

A Typeful Characterization of Multiparty Structured Conversations Based on Binary Sessions

Luís Caires

Universidade Nova de Lisboa
lcaires@fct.unl.pt

Jorge A. Pérez

University of Groningen
j.a.perez@rug.nl

Abstract

Relating the specification of the global communication behavior of a distributed system and the specifications of the local communication behavior of each of its nodes/peers (e.g., to check if the former is realizable by the latter under some safety and/or liveness conditions) is a challenging problem addressed in many relevant scenarios. In the context of networked software services, a widespread programming language-based approach relies on global specifications defined by session types or behavioral contracts. Static type checking can then be used to ensure that components follow the prescribed interaction protocols. In the case of session types, developments have been mostly framed within quite different type theories for either *binary* (two-party) or *multiparty* (n -party) protocols. Unfortunately, the precise relationship between analysis techniques for multiparty and binary protocols is yet to be understood.

In this work, we bridge this previously open gap in a principled way: we show that the analysis of multiparty protocols can also be developed within a much simpler type theory for binary protocols, ensuring protocol fidelity and deadlock-freedom. We present characterization theorems which provide new insights on the relation between two existing, yet very differently motivated, session type systems—one based on linear logic, the other based on automata theory—and suggest useful type-based verification techniques for multiparty systems relying on reductions to the binary case.

Keywords Concurrency, Behavioral Types, Multiparty Communication, Linear Logic, Session Types, Process Calculi.

1. Introduction

Relating the global specification of a distributed system and the set of components that implement such a specification is a problem found in many relevant scenarios. For instance, the analysis of security protocol narrations relies on formal correspondences between a global protocol specification and implementations for the each roles/principals in the protocol (see, e.g., [20]). Also, the formal connection between Message Sequence Charts (MSCs) and Communicating Finite State Machines (CFSMs) has been thoroughly studied (see, e.g., [14]). A recent survey by Castagna et al. [8, §7] contrasts these two scenarios.

Establishing relationships between global and local specifications is also important in the analysis of networked software services. In this context, the emphasis is in the analysis of message-passing, communication-oriented programs, which often feature advanced forms of concurrency, distribution, and trustworthiness. Within programming language-based techniques, notable approaches include *interface contracts* (see, e.g., [7]) and *behavioral types* [1, 4, 6, 15–18]. Our focus is in the latter, often defined on top of core programming calculi (usually, some dialect of the π -calculus [22]) which specify the inherently interactive nature of communication-based systems. By classifying behaviors (rather

than values), behavioral types define abstract descriptions of structured protocols, and enforce disciplined exchanges of values and communication channels. Advantages of the behavioral types approach include simplicity and flexibility; successful integrations of behavioral types into functional and object-oriented languages [30] offer compelling evidence of these benefits. A variety of frameworks based on behavioral types has been put forward, revealing a rather rich landscape of models and languages in which communication is delineated by types. In particular, frameworks based on *session types* [15, 16] have received much attention in academic and applied contexts. In these models, multiparty conversations are organized as concurrent *sessions*, which define structured dialogues. Unfortunately, the diversity of underlying models and techniques makes formally relating independently defined typed frameworks a challenging task. This limitation makes it difficult to objectively compare the expressiveness and significance of seemingly related behavioral type theories. Also, it hinders the much desirable transfer of reasoning/verification techniques between different typed frameworks—a notable example being techniques for ensuring deadlock-free protocols (see below).

In this paper, we identify formal relationships between two distinct typed frameworks for structured communications. Precisely, we establish natural and useful bridges between typed models for *binary* and *multiparty* session communications [16, 17]: by relying on a theory of binary session types rooted on linear logic [3, 29], we elucidate new deep foundational connections between both frameworks. Our results not only reveal new logically motivated justifications for key concepts in models of typeful specifications of global and local behaviors; they also enable the principled transfer of useful reasoning techniques. As we argue next, this is rather significant for session types, as (well-understood) techniques in the binary setting are usually hard to generalize to multiparty sessions.

In binary communications [16] protocols involve exactly two partners, each one abstracted by a behavioral type; correct interactions depend on *compatibility*, which intuitively means that when one partner performs an action (e.g., send), the other performs a complementary one (e.g., receive). In the multiparty setting, protocols may involve an arbitrary number of partners. There is a global specification (or *choreography*) to which all partners, from their local perspectives, should adhere. In multiparty session types [11, 12, 17], these two visions are described by a *global type* and *local types*, respectively; a *projection function* formally relates the two. The expressiveness jump from binary to multiparty communications is significant. Extensive research has shown that type systems for multiparty communication have a much more involved theory than binary ones. In session types, e.g., this shows up in basic concepts such as compatibility: while binary compatibility can be simply characterized as *type duality* [16], a formal characterization of multiparty compatibility was given only recently [12]. Also, some analysis techniques, such as those for deadlock-freedom, are

a difficult issue in the multiparty setting, and certainly harder than in the binary case. The question that arises is then: could multiparty session types be reduced, in some precise sense, into binary ones? As discussed in [17], such a reduction entails decomposing a single global specification into several binary fragments. Defining such a decomposition is far from trivial, for it should satisfy at least two requirements. First, it must preserve crucial *sequencing information* among multiparty exchanges. Second, the resulting collection of binary interactions should not exhibit *undesirable behaviors*, such as, e.g., synchronization errors, unspecified protocol steps, deadlocks, and unproductive, non-terminating reductions.

This paper answers the above question in the affirmative. We exhibit a tight correspondence between:

- a standard theory of multiparty session types (as first formulated by Honda, Yoshida, and Carbone [17] and recently characterized via communicating automata by Deniérou and Yoshida [12]), and
- the theory of deadlock-free binary session types proposed by Caires and Pfenning in [3], which interprets linear logic propositions as session types, in the style of Curry-Howard.

We briefly motivate our approach. Deniérou and Yoshida have recently identified the set of global types that admit a sound and complete characterization of multiparty compatibility in terms of communicating automata [12]. This suggests whether multiparty communication could be related to binary communication. An initial observation is that a communication from p to q can be *decoupled* into two simpler steps: a *send* action from p to an intermediate entity, followed by a step in which the entity *forwards* the message to q . Our approach generalizes this observation: given a multiparty conversation G (a global type), we extract its semantics in terms of a *medium process*, denoted $M[G]$, an intermediate entity in all protocol exchanges (§ 4.1). Process $M[G]$ may uniformly capture all the sequencing information prescribed by G . The next step for a full characterization is determining the conditions under which $M[G]$ may be well-typed in a theory for binary session types, with respect to the local types for G (i.e., its associated projections). A key ingredient in our characterization is the theory for binary session types introduced in [3] and extended in [29]. Due to their logical foundations, well-typed processes are naturally type preserving; this entails *fidelity* (i.e., protocols are respected) and *safety* (i.e., absence of runtime communication errors). Moreover, well-typed processes are *deadlock-free* (i.e., processes do not get stuck) and *compositionally non-diverging* (i.e., infinite observable behavior is allowed, but unproductive, infinite internal behavior is ruled out). Particularly relevant for our approach is deadlock-freedom, not directly ensured by alternative type systems. Our core developments rely on tight relationships between global and binary session types:

- For any global type G which is *well-formed* (in the sense of Deniérou and Yoshida [12]), a medium process $M[G]$ can be constructed such that $M[G]$ is well-typed in the binary session type system of [3], under a typing environment in which participants are assigned binary types that correspond to the expected projections of G onto all participants.
- Conversely, for any G such that $M[G]$ is well-typed in the type system of [3, 29], under a typing environment assigning some binary session types for participants, such binary types correspond, in a very precise sense, to the projections of G .

Notice that (a) immediately provides an alternative proof of global progress/deadlock-freedom for global systems as in [12, 17]. Moreover, given the uniform definition of medium processes, our results immediately apply to more expressive systems, in particular supporting name passing, session delegation, and parallel composition of global types, all of these features being beyond the scope of [12]. It is also worth highlighting that, unlike, e.g., [12], our medium

characterization of multiparty session types sbridges all the way from global types to actual processes implementing the system. This allows us to explore known properties of the underlying typed processes, in particular, behavioral equivalences [25] (see below), to reason about (and justify) properties of multiparty systems.

Contributions. This paper offers the following contributions.

- We offer an analysis of multiparty session types using a theory of binary session types, ensuring fidelity and deadlock-freedom. We give a two-way correspondence relating well-formed global types with typability of its associated medium on a logically motivated theory of binary sessions. This results holds both for global types without recursion (but with parallel composition, cf. Theorems 4.8 and 4.10) and for global types with recursion (and without parallel composition, exactly the same type structure studied in [12], cf. Theorem 4.16 and 4.17).
- For global types without recursion (but with parallel composition), we show how known typed behavioral equivalences for binary sessions [25] may be used to justify behavioral transformations of global types, expressed at the level of mediums (Theorem 4.14). This provides a deep semantic justification of useful structural identities on global types, such as those capturing parallelism via commutation or interleaving of causally independent communications [6].
- For global types with recursion (and without parallel composition), we prove an operational correspondence result relating the behavior of a global type with the observable behavior of the composition of (a) its medium process (instrumented in a natural way) and (b) arbitrary implementations for local participants (Theorem 4.23). This confirms that mediums faithfully mirror global specifications. That is, going through the medium does not introduce extra sequentiality in protocols, as mediums are essentially transparent from an operational standpoint.
- We show how an approach based on mediums allows to effectively transfer techniques from binary sessions to multiparty session types. We show how to enrich global type specifications with an expressive *join primitive*. Also, we describe how to analyze global types with *parametric polymorphism* based on already existing theories in the binary setting (§ 6). The latter is remarkable, for we do not know of multiparty session type theories supporting parametric polymorphism.

Our results not only clarify the relationship between multiparty and binary session types. They also give further evidence on the fundamental character of the notions involved (projection functions, type primitives, multiparty compatibility), since they can be independently and precisely explained within the separate worlds of communicating automata and linear logic.

Next we illustrate our approach and key results by means of an example. § 3 collects definitions on multiparty and binary sessions. Our main results are reported in § 4. Further illustration is given in § 5. In § 6 we describe extensions to our approach, while § 7 discusses related works. § 8 collects some concluding remarks.

An accompanying appendix gives details of omitted definitions and proofs.

2. Our Approach and Main Results, By Example

We now elaborate further on our contributions by illustrating how our approach bridges all the way from global types (choreographies) to actual π -calculus processes which realize multiparty scenarios and inherit key properties from binary session typing, most notably, protocol fidelity and freedom from deadlocks.

We illustrate the multiparty session types approach by an example. In this paper, we consider the following syntax of global types:

$G ::=$	$p \rightarrow q : \{1_i(U_i).G_i\}_{i \in I}$	communication from p to q
	$G_1 \mid G_2$	composition of global types
	$\mu \mathcal{X}. G \mid \mathcal{X}$	recursive global type
	end	terminated global type

Consider global type G_c below, which abstracts a variant of the *commit protocol* in [12]. The protocol involves three participants: A, B, and C. First, A orders to B to *act* or to *quit*. In the first case, B sends a *signal* to C and subsequently A sends to C a *commit* order. In the second case, B orders to C to *save*, then A orders C to *finish*:

$$G_c = A \rightarrow B : \{ \text{act}(\text{int}).B \rightarrow C : \{ \text{sig}(\text{str}).A \rightarrow C : \{ \text{comm}(\text{1}).\text{end} \} \}, \\ \text{quit}(\text{int}).B \rightarrow C : \{ \text{save}(\text{1}).A \rightarrow C : \{ \text{fini}(\text{1}).\text{end} \} \} \}$$

Given a global specification such as G_c , to derive code for each participant in the choreography, one first extracts a set of so-called *local types*. Formally, these local types are obtained using a *projection function* (cf. Def. 3.3). This way, e.g., the local behavior for A and C, denoted $G_c \upharpoonright A$ and $G_c \upharpoonright C$, is given by the local types:

$$G_c \upharpoonright A = A! \{ \text{act}(\text{int}).A! \{ \text{comm}(\text{1}).\text{end} \}, \\ \text{quit}(\text{int}).B! \{ \text{sig}(\text{str}).\text{end} \} \} \\ G_c \upharpoonright C = B? \{ \text{sig}(\text{str}).A? \{ \text{comm}(\text{1}).\text{end} \}, \\ \text{save}(\text{1}).A? \{ \text{fini}(\text{1}).\text{end} \} \}$$

The local type $G_c \upharpoonright B$ is obtained similarly. Not all global types are meaningful: following [12], we focus on *well-formed global types*—global types which have a well-defined local type for each participant. For well-formed global types, one may independently obtain implementations for each participant; then, using associated type systems [17], it can be ensured that such implementations are type-safe and realize the intended global scenario. Using complementary techniques one may guarantee other advanced properties, such as deadlock-freedom in interleaved sessions (cf. [9]). Also, using the correspondence with communicating automata [12], safety and liveness guarantees carry over to choreographies.

In this paper, we propose analyzing global types by means of an existing type system for binary sessions, which defines a Curry-Howard isomorphism between intuitionistic linear logic propositions and session types [3, 29]. The analysis that we propose is thus endowed with a deep foundational significance. The main conceptual device in our approach is the *medium process* of a global type (or simply *medium*). Extracted following the syntax of the global type, a medium process is intended to interact with all the implementations which conform to the local types obtained via projection. Before describing the medium for G_c above, let us give some intuitions on our process syntax—a standard π -calculus with n -ary labeled choice (cf. Def. 3.5). We write $a(v)$ and $\bar{b}(w)$ to denote, respectively, prefixes for *input* (along name a , with v being a placeholder) and *bound output* (along name b , with w being a freshly generated name). Labeled choice is specified using processes $a \triangleright \{ \text{label}_1 : P_1, \dots, \text{label}_n : P_n \}$ (branching) and $a \triangleleft \text{label}_1 ; Q$ (selection). The interaction of input and output prefixes (resp. branching and selection constructs) defines a *reduction*. Process $[u \leftrightarrow y]$ equates names u and y , while $(\nu x)P$ and $P \mid Q$ denote the usual restriction and parallel composition constructs.

The medium of a global type is a process in which every directed labeled communication is captured by its decomposition into simpler prefixes. This way, e.g. the medium of G_c , denoted $M[G_c]$, is as in Fig. 1, where we have used names a , b , and c to indicate interactions associated to A, B, and C, respectively. With this in mind, we may now informally explain how the medium $M[G_c]$ gives semantics to the global type G_c . Consider the first labeled communication in G_c , in which A sends a value to B by selecting a label *act* or *quit*. We assume that process implementations for A and

$$a \triangleright \{ \text{act} : a(v).b \triangleleft \text{act}; \bar{b}(w).([w \leftrightarrow v] \mid \\ b \triangleright \{ \text{sig} : b(n).c \triangleleft \text{sig}; \bar{c}(m).([n \leftrightarrow m] \mid \\ a \triangleright \{ \text{comm} : a(u).c \triangleleft \text{comm}; \bar{c}(y).([u \leftrightarrow y] \mid \mathbf{0}) \} \}) \}, \\ \text{quit} : a(v).b \triangleleft \text{quit}; \bar{b}(w).([w \leftrightarrow v] \mid \\ b \triangleright \{ \text{save} : b(n).c \triangleleft \text{save}; \bar{c}(m).([n \leftrightarrow m] \mid \\ a \triangleright \{ \text{fini} : a(u).c \triangleleft \text{fini}; \bar{c}(y).([u \leftrightarrow y] \mid \mathbf{0}) \} \}) \} \}$$

Figure 1. Medium process $M[G_c]$ for global type G_c

B are available on names a and b , respectively. The implementation for A should first select a label (say *act*) and then output a message for B (say, a name y denoting a reference to an integer). Accordingly, the first two actions of $M[G_c]$ are on name a : it first commits to the labeled alternative *act* and then it receives y . This completes the involvement of A in the communication to B. Subsequently, the medium acts on name b , first selecting label *act* in the implementation of B and then sending a fresh name w . Then, names w and y (i.e., the one received from A) are “linked” together, and the medium for the continuation of the global protocol is spawned.

Mediums give a simple characterization of global types; intuitively, they define the code for “gluing together” the behavior of all local participants. However, a medium by itself does not relate a global type and the local types obtained by projection. Giving a logically motivated justification for this connection is the central contribution of this paper. To this end, we rely on the theory of binary session types developed in [3, 29], which connects intuitionistic linear logic propositions and session types. This way, e.g., session types $!A; B$ and $?A; B$, introduced in [16] to abstract input and output, are represented in [3, 29] by the tensor $A \otimes B$ and the linear arrow $A \multimap B$. More precisely, we have:

$A, B, C ::=$	$\mathbf{1}$	terminated binary session
	$A \multimap B$	$A \otimes B$ input, output
	$A \& B$	$A \oplus B$ branching and selection
	$!A \mid \nu \mathcal{X}. A \mid \mathcal{X}$	replicated and coinductive type

We use the expected extension of the binary operators $\&$ and \oplus to the n -ary case. Type assignments are of the form $x:A$, where x is a name and A a session type. Given a process P , the typing judgment

$$\Gamma; \Delta \vdash_\eta P :: z:C$$

asserts that P provides a behavior described by session type C at channel/name z , building on *linear* session behaviors declared in type environment Δ and *unrestricted* (or *persistent*) behaviors declared in type environment Γ . The map η relates (corecursive) type variables to typing contexts [29]. Thus, judgments in [3, 29] specify both the behavior that a process *offers* (or *implements*) and the (unrestricted, linear) behaviors that it *requires* to do so. As discussed above, well-typed processes in this system are naturally type-preserving and deadlock-free. They are also non-diverging.

Our characterization results concern well-typedness of mediums in the logic-based theory of binary sessions. In fact, Theorems 4.8 and 4.10 ensure that the medium process $M[G_c]$ given above is typable as follows:

$$\cdot; a:\langle\langle G_c \upharpoonright A \rangle\rangle, b:\langle\langle G_c \upharpoonright B \rangle\rangle, c:\langle\langle G_c \upharpoonright C \rangle\rangle \vdash_\eta M[G_c] :: z:\mathbf{1} \quad (1)$$

where \cdot stands for the empty (shared) environment and $\langle\langle \cdot \rangle\rangle$ relates local types and binary session types (cf. Def. 4.7). Well-typedness of $M[G_c]$ as captured by (1) has significant consequences:

- It formalizes the dependence of the medium on behaviors with local types defined by projection: (1) says that $M[G_c]$ requires exactly behaviors $\langle\langle G_c \upharpoonright A \rangle\rangle$, $\langle\langle G_c \upharpoonright B \rangle\rangle$, and $\langle\langle G_c \upharpoonright C \rangle\rangle$, which should be available, as linear resources, along names a , b , and c .

- b. The judgement also ensures that the medium does not add extraneous behaviors: since the offered behavior of $M[G_c]$ (in the right-hand side) is 1 , we are sure that it acts as a faithful mediator among the behaviors described in the left-hand side.
- c. By well-typedness, $M[G_c]$ inherits type preservation, deadlock-freedom, and non-divergence as ensured by the type system in [3, 29]. This not only means that $M[G_c]$ in isolation behaves as intended. Consider processes P_A , Q_B , and R_C which implement $\langle\langle G_c \mid A \rangle\rangle$, $\langle\langle G_c \mid B \rangle\rangle$, and $\langle\langle G_c \mid C \rangle\rangle$ in appropriate names:

$$\begin{aligned} \cdot; \cdot \vdash_{\eta} P_A &:: a:\langle\langle G_c \mid A \rangle\rangle & \cdot; \cdot \vdash_{\eta} Q_B &:: b:\langle\langle G_c \mid B \rangle\rangle \\ \cdot; \cdot \vdash_{\eta} R_C &:: c:\langle\langle G_c \mid C \rangle\rangle \end{aligned}$$

where, for simplicity, we have assumed no linear/shared dependencies. These processes can be constructed independently from (and unaware of) $M[G_c]$, and type-checked in the system of [3, 29], inheriting all key properties. Moreover, the composition of $M[G_c]$ with such processes (i.e., a *system* realizing G_c)

$$(\nu c)((\nu b)((\nu a)(M[G_c] \mid P_A) \mid Q_B) \mid R_C)$$

is also well-typed, and therefore type-preserving, deadlock-free, and non-diverging. Deadlock-freedom can be seen to be crucial in ensuring faithful sets of interactions between the local implementations P_A , Q_B , R_C and the medium $M[G_c]$.

To further support point (b) above, we define the *annotated medium* of G_c , denoted $\mathcal{M}^{\mu}[G_c]_k$ (cf. Def. 4.4). This process extends $M[G_c]$ with visible actions on name k which mimic those in G_c . Our main results extend smoothly to annotated mediums, and we may derive the following:

$$\cdot; a:\langle\langle G_c \mid A \rangle\rangle, b:\langle\langle G_c \mid B \rangle\rangle, c:\langle\langle G_c \mid C \rangle\rangle \vdash_{\eta} \mathcal{M}^{\mu}[G_c]_k :: k:\langle\langle G_c \rangle\rangle \quad (2)$$

where $\langle\langle G_c \rangle\rangle$ denotes a session type which captures the sequentiality of G_c (cf. Def. 4.18). Building upon (2), we may state a rather strong result of operational correspondence relating global types and annotated mediums, which is given by Theorem 4.23. Roughly speaking, such a result identifies the conditions under which each step of type G_c (as formalized by a simple LTS for global types) can be matched by a labeled transition of process $\mathcal{M}^{\mu}[G_c]_k$.

3. Preliminaries: Multiparty and Binary Sessions

Here we collect key definitions for multiparty session types, as presented in [12, 17] (§3.1). We also summarize the key concepts of the logically motivated theory of binary sessions, as first introduced in [3] and extended with coinductive session types in [29] (§3.2).

3.1 Multiparty Session Types

Our syntax of global and local types subsumes constructs from the original presentation [17] and from recent formulations [12]. With respect to [17], we consider labeled communication, recursion, and retain parallel composition, which enables compositional reasoning over global specifications. With respect to [12], we consider value/session passing in branching (cf. U below) and add parallel composition. Below, *participants* and *labels* are ranged over by p, q, r, \dots and l_1, l_2, \dots , respectively.

Definition 3.1 (Global and Local Types). *Define global and local types as*

$$\begin{aligned} G &::= \text{end} \mid p \rightarrow q: \{l_i \langle U_i \rangle. G_i\}_{i \in I} \mid G_1 \mid G_2 \mid \mu \mathcal{X}. G \mid \mathcal{X} \\ U &::= \text{bool} \mid \text{nat} \mid \text{str} \mid \dots \mid T \\ T &::= \text{end} \mid p? \{l_i \langle U_i \rangle. T_i\}_{i \in I} \mid p! \{l_i \langle U_i \rangle. T_i\}_{i \in I} \mid \mu \mathcal{X}. T \mid \mathcal{X} \end{aligned}$$

The set of participants of G , denoted $\text{part}(G)$, is defined as: $\text{part}(p \rightarrow q: \{l_i \langle U_i \rangle. G_i\}_{i \in I}) = \{p, q\} \cup \bigcup_{i \in I} \text{part}(G_i)$,

$\text{part}(G_1 \mid G_2) = \text{part}(G_1) \cup \text{part}(G_2)$, $\text{part}(\mu \mathcal{X}. G) = \text{part}(G)$, $\text{part}(\text{end}) = \text{part}(\mathcal{X}) = \emptyset$. We sometimes write $r \in G$ to abbreviate $r \in \text{part}(G)$.

The global type $p \rightarrow q: \{l_i \langle U_i \rangle. G_i\}_{i \in I}$ specifies that, by choosing label l_i , p may send to q a message of type U_i , with subsequent behavior G_i . As in [12, 17], we decree $p \neq q$, so reflexive interactions are disallowed. Also, set I is finite and labels are assumed pairwise different. The global type $G_1 \mid G_2$, introduced in [17], allows the concurrent execution of G_1 and G_2 . Global type $\mu \mathcal{X}. G$ defines recurring conversation structures. As in [12, 17], we restrict to global recursive types in which type variables \mathcal{X} are all *guarded*, i.e., they may only occur under branchings. Global type end denotes the completed choreography. At the local level, branching types $p? \{l_i \langle U_i \rangle. T_i\}_{i \in I}$ and selection types $p! \{l_i \langle U_i \rangle. T_i\}_{i \in I}$ have expected readings. The terminated local type is denoted end .

We now define *projection* and *well-formedness* for global types. We consider *merge-based* projection as first proposed in [13] and used in [12]. The definition below adds flexibility to the one in [17] by relying on a *merge* operator on local types; we give a simpler presentation of the definition in [12, §3], considering messages U .

Definition 3.2 (Merge). *We define \sqcup as the commutative partial operator on base and local types such that:*

1. $\text{bool} \sqcup \text{bool} = \text{bool}$ (and analogously for other base types)
2. $\text{end} \sqcup \text{end} = \text{end}$
3. $p! \{l_i \langle U_i \rangle. T_i\}_{i \in I} \sqcup p! \{l_i \langle U_i \rangle. T_i\}_{i \in I} = p! \{l_i \langle U_i \rangle. T_i\}_{i \in I}$
4. $p? \{l_k \langle U_k \rangle. T_k\}_{k \in K} \sqcup p? \{l'_j \langle U'_j \rangle. T'_j\}_{j \in J} = p? (\{l_k \langle U_k \rangle. T_k\}_{k \in K} \cup \{l'_j \langle U'_j \rangle. T'_j\}_{j \in J})$ if for all $k \in K, j \in J$, $(l_k = l'_j)$ implies that both $U_k \sqcup U'_j$ and $T_k \sqcup T'_j$ are defined.
5. $\mathcal{X} \sqcup \mathcal{X} = \mathcal{X}$ and $\mu \mathcal{X}. G_1 \sqcup \mu \mathcal{X}. G_2 = \mu \mathcal{X}. (G_1 \sqcup G_2)$.

and is undefined otherwise.

Intuitively, for $U_1 \sqcup U_2$ to be defined there are two options: (a) U_1 and U_2 are both identical base, terminated or selection types; (b) U_1 and U_2 are both branching types, but not necessarily identical: they may offer different options but with the condition that the behavior in labels occurring in both U_1 and U_2 must be the same, up to \sqcup .

Definition 3.3 (Projection [12, 13]). *Let G be a global type. The (merge-based) projection of G under participant r , denoted $G \upharpoonright r$, is defined as follows:*

- $\text{end} \upharpoonright r = \text{end}$
- $p \rightarrow q: \{l_i \langle U_i \rangle. G_i\}_{i \in I} \upharpoonright r = \begin{cases} p! \{l_i \langle U_i \rangle. G_i \upharpoonright r\}_{i \in I} & \text{if } r = p \\ p? \{l_i \langle U_i \rangle. G_i \upharpoonright r\}_{i \in I} & \text{if } r = q \\ \bigcup_{i \in I} G_i \upharpoonright r & \text{otherwise, with } \sqcup \text{ as in Def. 3.2} \end{cases}$
- $(G_1 \mid G_2) \upharpoonright r = \begin{cases} G_i \upharpoonright r & \text{if } r \in G_i \text{ and } r \notin G_j, i \neq j \in \{1, 2\} \\ \text{end} & \text{if } r \notin G_1 \text{ and } r \notin G_2 \end{cases}$
- $\mathcal{X} \upharpoonright r = \mathcal{X}$
- $\mu \mathcal{X}. G \upharpoonright r = \begin{cases} \mu \mathcal{X}. G \upharpoonright r & \text{if } G \upharpoonright r \neq \mathcal{X} \\ \text{end} & \text{otherwise} \end{cases}$

When a side condition does not hold, the map is undefined.

Definition 3.4 (Well-Formed Global Types [12]). *We say global type G is well-formed (WF, in the following) if for all $r \in G$, the projection $G \upharpoonright r$ is defined.*

3.2 Binary Session Types Based on Linear Logic

In this paper we build upon the theory of binary session types of [3, 29], based on a Curry-Howard interpretation of session types as linear logic propositions. In the remainder, we assume no special background on linear logic; we refer to [3, 29] for further details.

$$\begin{aligned}
& x(y).Q \mid x(z).P \rightarrow Q \mid P\{y/z\} \\
& x(y).Q \mid !x(z).P \rightarrow Q \mid P\{y/z\} \mid !x(z).P \\
& (\nu x)([x \leftrightarrow y] \mid P) \rightarrow P\{y/x\} \\
& Q \rightarrow Q' \Rightarrow P \mid Q \rightarrow P \mid Q' \\
& P \rightarrow Q \Rightarrow (\nu y)P \rightarrow (\nu y)Q \\
& P \equiv P', P' \rightarrow Q', Q' \equiv Q \Rightarrow P \rightarrow Q \\
& x \triangleleft 1_j; P \mid x \triangleright \{1_i : Q_i\}_{i \in I} \rightarrow P \mid Q_j \quad (j \in I) \\
& (\text{corec } \mathcal{X}(\tilde{y}).P) \tilde{c} \rightarrow P\{\tilde{c}/\tilde{y}\} \{ \text{corec } \mathcal{X}(\tilde{y}).P / \mathcal{X} \}
\end{aligned}$$

Figure 2. Process Reduction

The Process Model: Syntax and Semantics. We define a synchronous π -calculus [27] with replication, forwarding, n -ary labeled choice, and co-recursive definitions/variables (cf. [29]). As for global types, we use $1_1, 1_2, \dots$ to range over *labels*.

Definition 3.5 (Processes). *Given an infinite set Λ of names (x, y, z, u, v) , the set of processes (P, Q, R) is defined by*

$$\begin{array}{l|l|l}
P ::= & \mathbf{0} & P \mid Q \\
& \bar{x}y.P & x(y).P \\
& x \triangleleft 1_i; P & x \triangleright \{1_i : P_i\}_{i \in I} \\
& \mathcal{X}(\tilde{c}) & (\text{corec } \mathcal{X}(\tilde{y}).P) \tilde{c}
\end{array}
\quad
\begin{array}{l}
(\nu y)P \\
!x(y).P \\
[x \leftrightarrow y]
\end{array}$$

The operators $\mathbf{0}$ (inaction), $P \mid Q$ (parallel composition), and $(\nu y)P$ (name restriction) are standard. We then have $\bar{x}y.P$ (send y on x and proceed as P), $x(y).P$ (receive a z on x and proceed as P with parameter y replaced by z), and $!x(y).P$ which denotes replicated (persistent) input. The forwarding construct $[x \leftrightarrow y]$ equates names x and y ; it is a primitive representation of a copycat process. The operators $x \triangleleft 1_i; P$ and $x \triangleright \{1_i : P_i\}_{i \in I}$ define labeled choice as in [16]. Given a sequence of names \tilde{c} , constructs $(\text{corec } \mathcal{X}(\tilde{y}).P) \tilde{c}$ and $\mathcal{X}(\tilde{c})$ represent co-recursive definitions and co-recursive variables, respectively. We write $\bar{x}(y)$ to stand for $(\nu y)\bar{x}y$.

In restriction $(\nu y)P$ and input $x(y).P$ the distinguished occurrence of name y is binding, with scope P . The set of *free names* of a process P is denoted $fn(P)$. A process is *closed* if it does not contain free occurrences of names. We identify processes up to consistent renaming of bound names, writing \equiv_α for this congruence. We write $P\{x/y\}$ for the capture-avoiding substitution of x for y in P . *Structural congruence* ($P \equiv Q$) expresses basic identities on the structure of processes. It is defined as the least congruence relation on processes such that

$$\begin{aligned}
P \mid \mathbf{0} &\equiv P & P \mid Q &\equiv Q \mid P & P \mid (Q \mid R) &\equiv (P \mid Q) \mid R \\
(\nu x)(\nu y)P &\equiv (\nu y)(\nu x)P & (\nu x)\mathbf{0} &\equiv \mathbf{0} & P \equiv_\alpha Q &\Rightarrow P \equiv Q \\
[x \leftrightarrow y] &\equiv [y \leftrightarrow x] & x \notin fn(P) &\Rightarrow P \mid (\nu x)Q &\equiv (\nu x)(P \mid Q)
\end{aligned}$$

Reduction specifies the computations a process performs on its own. Closed under structural congruence, it is the binary relation on processes defined by the rules in Fig. 2.

Session Types as Linear Logic Propositions. The theory of binary session types of [3] connects session types as linear logic propositions. This correspondence has been extended in [29] with coinductive session types. Main properties derived from typing, important for our work and absent from other binary session type theories are *global progress* and *compositional non-divergence*.

Definition 3.6 (Binary Types). *Types (A, B, C) are given by*

$$\begin{array}{l}
A, B ::= \mathbf{1} \mid !A \mid A \otimes B \mid A \multimap B \mid \mathcal{X} \mid \nu \mathcal{X}.A \\
\quad \mid \&\{1_i : A_i\}_{i \in I} \mid \oplus \{1_i : A_i\}_{i \in I}
\end{array}$$

Types are assigned to names; assignment $x:A$ enforces the use of x according to discipline A . As hinted at above, we use $A \otimes B$ (resp. $A \multimap B$) to type a name that performs an output (resp. an input) to its partner, sending (resp. receiving) a name of type A , and then behaves as type B . We generalize the type syntax in [3] by

considering n -ary offer $\&$ and choice \oplus . Given a finite index set I , $\&\{1_i : A_i\}_{i \in I}$ types a name that offers a choice between an l_i . Dually, $\oplus \{1_i : A_i\}_{i \in I}$ types the selection of one of the 1_i . Type $!A$ types a shared channel, to be used by a server for spawning an arbitrary number of new sessions (possibly none), each one conforming to type A . We use $\nu \mathcal{X}.A$ to type coinductive sessions, here required to type the medium processes of global types with recursion. Coinductive types are required to have strictly positive occurrences of the type variable. Also, coinductive types without session behavior before the occurrence of the type variable (e.g., $\nu \mathcal{X}. \mathcal{X}$) are excluded. Finally, type $\mathbf{1}$ means that the session terminated, no further interaction will take place on it; names of type $\mathbf{1}$ may be passed around in sessions, as opaque values.

A *type environment* collects type assignments of the form $x:A$, where x is a name and A a type, the names being pairwise disjoint. We consider two typing environments, subject to different structural properties: a *linear* part Δ and an *unrestricted* part Γ , where weakening and contraction principles hold for Γ but not for Δ .

A *type judgment* is of the form $\Gamma; \Delta \vdash_\eta P :: z:C$. It asserts that P provides behavior C at channel z , building on “services” declared in $\Gamma; \Delta$. Recall that η denotes a map from (corecursive) type variables to typing contexts. The domains of Γ, Δ and $z:C$ are required to be pairwise disjoint. As π -calculus terms are considered up to structural congruence, typability is closed under \equiv by definition. As a simple example of type judgment, a client Q that relies on external services and does not provide any is typed as $\Gamma; \Delta \vdash Q :: z:\mathbf{1}$. Empty environments Γ, Δ are denoted by ‘ \cdot ’. Also, we sometimes abbreviate $\Gamma; \Delta \vdash_\eta P :: z:\mathbf{1}$ as $\Gamma; \Delta \vdash P$.

Fig. 3 presents selected rules of the type system; see [3, 29] for a full account. Due to the logic correspondence, we have right (R) and left (L) rules. The former detail how a process can implement the behavior described by the given connective; the latter explain how a process may use of a session of a given type. Given these intuitions, the interpretation of the various rules should be clear. Rule (Tid) defines identity in terms of forwarding. Rule (Tcut) define typed composition, restricting the scope of involved processes. Based on rules (Tcut) and (T1L), a rule for *independent parallel composition*, enabling the composition of $\Gamma; \Delta_1 \vdash P :: z:\mathbf{1}$ (with $z \notin fn(P)$, cf. rule (T1R)) and $\Gamma; \Delta_2 \vdash Q :: x:A$ into $\Gamma; \Delta_1, \Delta_2 \vdash P \mid Q :: x:A$ is derivable. Implementing a session with type $\&\{1_i : A_i\}_{i \in I}$ amounts to offering a choice between n sessions with type A_i (cf. rule (T&R)). Using a session of type $\&\{1_i : A_i\}_{i \in I}$ on name x entails selecting one of the alternatives, using a prefix $x \triangleleft 1_j$ (cf. rules (T&L₁) and (T&L₂)). The interpretation for the n -ary additive disjunction $\oplus \{1_i : A_i\}_{i \in I}$ is dual.

We now recall some main results for well-typed processes. For any P , define *live*(P) iff $P \equiv (\nu \tilde{n})(\pi.Q \mid R)$, for some names \tilde{n} , a process R , and a *non-replicated* guarded process $\pi.Q$. Also, we write $P \Downarrow$, if there is no infinite reduction path from process P .

Theorem 3.7 (Properties of Well-Typed Session Processes, [29]). *Suppose $\Gamma; \Delta \vdash_\eta P :: z:A$.*

1. *Type Preservation: If $P \rightarrow Q$ then $\Gamma; \Delta \vdash_\eta Q :: z:A$.*
2. *Progress: If *live*(P) then there is Q with $P \rightarrow Q$ or one of the following holds:*
 - (a) $\Delta = \Delta', y:B$, for some Δ' and $y:B$.
There exists $\Gamma; \Delta' \vdash_\eta R :: y:B$ s.t. $(\nu y)(R \mid P) \rightarrow Q$.
 - (b) Exists $\Gamma; z:A, \Delta' \vdash_\eta R :: w:C$ s.t. $(\nu z)(P \mid R) \rightarrow Q$.
 - (c) $\Gamma = \Gamma', u:B$, for some Γ' and $u:B$.
There exists $\Gamma; \cdot \vdash_\eta R :: x:B$ s.t. $(\nu u)(!u(x).R \mid P) \rightarrow Q$.
3. *Non-Divergence: $P \Downarrow$.*

In particular, Theorem 3.7(2), key in our developments, implies that our type discipline ensures freedom from deadlocks.

$$\begin{array}{c}
\text{(Tid)} \quad \frac{}{\Gamma; x:A \vdash_{\eta} [x \leftrightarrow z] :: z:A} \quad \text{(T1L)} \quad \frac{\Gamma; \Delta \vdash_{\eta} P :: z:C}{\Gamma; \Delta, x:1 \vdash_{\eta} P :: z:C} \quad \text{(T1R)} \quad \frac{}{\Gamma; \cdot \vdash_{\eta} 0 :: x:1} \quad \text{(Tcut)} \quad \frac{\Gamma; \Delta \vdash_{\eta} P :: x:A \quad \Gamma; \Delta', x:A \vdash_{\eta} Q :: z:C}{\Gamma; \Delta, \Delta' \vdash_{\eta} (\nu x)(P \mid Q) :: z:C} \\
\text{(T}\neg\text{L)} \quad \frac{\Gamma; \Delta \vdash_{\eta} P :: y:A \quad \Gamma; \Delta', x:B \vdash_{\eta} Q :: z:C}{\Gamma; \Delta, \Delta', x:A \multimap B \vdash_{\eta} \overline{x}(y).(P \mid Q) :: z:C} \quad \text{(T}\otimes\text{L)} \quad \frac{\Gamma; \Delta, y:A, x:B \vdash_{\eta} P :: z:C}{\Gamma; \Delta, x:A \otimes B \vdash_{\eta} x(y).P :: z:C} \\
\text{(T}\oplus\text{L)} \quad \frac{\Gamma; \Delta, x:A_1 \vdash_{\eta} P_1 :: z:C \quad \dots \quad \Gamma; \Delta, x:A_k \vdash_{\eta} P_k :: z:C \quad I = \{1, \dots, k\}}{\Gamma; \Delta, x: \oplus\{1_i : A_i\}_{i \in I} \vdash_{\eta} x \triangleright \{1_i : P_i\}_{i \in I} :: z:C} \\
\text{(T}\oplus\text{R}_1\text{)} \quad \frac{\Gamma; \Delta \vdash_{\eta} P :: x:A}{\Gamma; \Delta \vdash_{\eta} x \triangleleft 1_i; P :: x: \oplus\{1_i : A\}_{i \in I}} \quad \text{(T}\oplus\text{R}_2\text{)} \quad \frac{\Gamma; \Delta \vdash_{\eta} P :: x: \oplus\{1_i : A_i\}_{i \in I} \quad k \notin I}{\Gamma; \Delta \vdash_{\eta} P :: x: \oplus\{1_j : A_j\}_{j \in I \cup \{k\}}} \quad \text{(T}\&\text{L}_1\text{)} \quad \frac{\Gamma; \Delta, x:A \vdash_{\eta} P :: z:C}{\Gamma; \Delta, x: \&\{1_i : A\}_{i \in I} \vdash_{\eta} x \triangleleft 1_i; P :: z:C} \\
\text{(T}\&\text{R)} \quad \frac{\Gamma; \Delta \vdash_{\eta} P_1 :: x:A_1 \quad \dots \quad \Gamma; \Delta \vdash_{\eta} P_k :: x:A_k \quad I = \{1, \dots, k\}}{\Gamma; \Delta \vdash_{\eta} x \triangleright \{1_i : P_i\}_{i \in I} :: x: \&\{1_i : A_i\}_{i \in I}} \quad \text{(T}\&\text{L}_2\text{)} \quad \frac{\Gamma; \Delta, x: \&\{1_i : A_i\}_{i \in I} \vdash_{\eta} P :: z:C \quad k \notin I}{\Gamma; \Delta, x: \&\{1_j : A_j\}_{j \in I \cup \{k\}} \vdash_{\eta} P :: z:C} \\
\text{(\nu L)} \quad \frac{\Gamma; \Delta, c:A\{\nu \mathcal{X}.A/\mathcal{X}\} \vdash_{\eta} Q :: d:D}{\Gamma; \Delta, c:\nu \mathcal{X}.A \vdash_{\eta} Q :: d:D} \quad \text{(\nu R)} \quad \frac{\Gamma; \Delta \vdash_{\eta'} P :: c:A \quad \eta' = \eta[\mathcal{X}(\tilde{y}) \mapsto \Gamma; \Delta \vdash_{\eta} c:\mathcal{Y}]}{\Gamma; \Delta \vdash_{\eta} (\text{corec } \mathcal{X}(\tilde{y}).P\{\tilde{y}/\tilde{z}\}) \tilde{z} :: c:\nu \mathcal{Y}.A} \quad \text{(VAR)} \quad \frac{\eta(\mathcal{X}(\tilde{y})) = \Gamma; \Delta \vdash_{\eta} d:\mathcal{Y} \quad \rho = \{\tilde{z}/\tilde{y}\}}{\rho(\Gamma); \rho(\Delta) \vdash_{\eta} \mathcal{X}(\tilde{z}) :: \rho(d):\mathcal{Y}}
\end{array}$$

Figure 3. The Type System for Binary Sessions: Selected Rules.

4. Relating Multiparty Protocols and Binary Session Typed Processes

Our typeful characterization of multiparty conversations as binary session types relies on the *medium process* of a global type. Mediums provide a simple conceptual device for analyzing global types using the logically motivated binary session types of [3, 29]. In fact, as the medium takes part in all message exchanges between local participants, it uniformly and cleanly captures the sequencing behavior stipulated by the global type.

For technical convenience, we divide the presentation of our main results into two representative sub-languages of global types:

- We shall write \mathcal{G}^{fin} to denote the class of global types generated by the syntax in Definition 3.1 *without recursion*.
- We shall write \mathcal{G}^{μ} to denote the class of global types generated by the syntax in Definition 3.1 *without the composition operator*.

Global types in \mathcal{G}^{fin} describe *finite choreographies*. Isolating this class is useful to illustrate the simplicity of our approach; in particular, to illustrate the fact that it is fully orthogonal from infinite behaviors induced by recursion. Investigating global types in \mathcal{G}^{μ} is insightful: this is exactly the class of global types for which Deniérou and Yoshida discovered the sound and complete characterization as communicating automata [12]. In §4.4 we discuss further the tension between composition and recursion in process characterizations of global types.

4.1 Medium Processes

Based on the distinction between \mathcal{G}^{fin} and \mathcal{G}^{μ} , we now introduce different definitions of medium processes. They all realize the simple concept motivated above and provide the basis for developing our technical results:

- For $G \in \mathcal{G}^{\text{fin}}$, we define *finite mediums* $M[G]$ (Def. 4.1) and establish characterization results connecting well-typed mediums and well-formed global types (Theorems 4.8 and 4.10). Based on finite mediums we also offer a behavioral characterization of *swapping* in global types (Theorem 4.14).
- For $G \in \mathcal{G}^{\mu}$, we define *recursive mediums* $M^{\mu}[G]_k$ (Def. 4.2) to extend the characterization results to global types featuring infinite behavior (Theorem 4.16 and 4.17). Then, we define *annotated mediums* $\mathcal{M}^{\mu}[G]_k$ (Definition 4.4). This notion, a slight

variation of Def. 4.2, allows us to precisely relate actions of the global type and the observable behavior of its associated annotated medium (Theorem 4.23).

We now proceed to define each of these different representations of the behavior of a global type.

Definition 4.1 (Finite Mediums). *Let $G \in \mathcal{G}^{\text{fin}}$ be a global type. The finite medium process of G , denoted $M[G]$, is defined inductively as follows:*

- $M[\text{end}] = 0$
- $M[p \multimap q: \{1_i \langle U_i \rangle . G_i\}_{i \in I}] = c_p \triangleright \{1_i : c_p(u).c_q \triangleleft 1_i; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G_i])\}_{i \in I}$
- $M[G_1 \mid G_2] = M[G_1] \mid M[G_2]$

We now introduce recursive mediums, which represent recursive global types using co-recursive processes. For technical reasons related to typability, we find it useful to “signal” when the global type ends and recurses. For simplicity, this signal is represented as a selection prefix (of a carried label) on a fresh name k :

Definition 4.2 (Recursive Mediums). *Let $G \in \mathcal{G}^{\mu}$ be a global type. Also, let k be a name assumed distinct from any other name. The recursive medium of G with respect to label 1, denoted $M^{\mu}[G]_k^1$, is defined as follows:*

- $M^{\mu}[\text{end}]_k^1 = k \triangleleft 1; 0$
- $M^{\mu}[p \multimap q: \{1_i \langle U_i \rangle . G_i\}_{i \in I}]_k^1 = c_p \triangleright \{1_i : c_p(u).c_q \triangleleft 1_i; \overline{c_q}(v).([u \leftrightarrow v] \mid M^{\mu}[G_i]_k^1)\}_{i \in I}$
- $M^{\mu}[\mu \mathcal{X}.G]_k^1 = (\text{corec } \mathcal{X}(\tilde{z}).M^{\mu}[G]_k^1) \tilde{c}$
- $M^{\mu}[\mathcal{X}]_k^1 = k \triangleleft 1; \mathcal{X}(\tilde{z})$

Let $1b$ be a label not in G . The recursive medium of G , denoted $M^{\mu}[G]_k$, is defined as $M^{\mu}[G]_k^{1b}$.

Example 4.3. *Finite mediums have already been illustrated in §2. To illustrate recursive mediums, consider the following variant of the commit protocol in §2. This is the running example in [12], here extended with base types:*

$$G_{rc} = \mu \mathcal{X}. A \multimap B: \{ \text{act} \langle \text{int} \rangle . B \multimap C: \{ \text{sig} \langle \text{str} \rangle . A \multimap C: \{ \text{comm} \langle 1 \rangle . \mathcal{X} \} \}, \text{quit} \langle \text{int} \rangle . B \multimap C: \{ \text{save} \langle 1 \rangle . A \multimap C: \{ \text{fini} \langle 1 \rangle . \text{end} \} \} \}$$

$$\begin{aligned} \mathbf{M}^\mu \llbracket G_{rc} \rrbracket_k = & \text{corec } \mathcal{X}(\tilde{z}). (a \triangleright \{ \text{act} : a(v).b \triangleleft \text{act}; \bar{b}(w).([w \leftrightarrow v] \mid b \triangleright \{ \text{sig} : b(n).c \triangleleft \text{sig}; \bar{c}(m). \\ & ([n \leftrightarrow m] \mid a \triangleright \{ \text{comm} : a(u).c \triangleleft \text{comm}; \bar{c}(y).([u \leftrightarrow y] \mid k \triangleleft \text{comm}; \mathcal{X}(\tilde{z})) \}) \}) \} , \\ & \text{quit} : a(v).b \triangleleft \text{quit}; \bar{b}(w).([w \leftrightarrow v] \mid b \triangleright \{ \text{save} : b(n).c \triangleleft \text{save}; \bar{c}(m). \\ & ([n \leftrightarrow m] \mid a \triangleright \{ \text{fini} : a(u).c \triangleleft \text{fini}; \bar{c}(y).([u \leftrightarrow y] \mid k \triangleleft \text{fini}; \mathbf{0}) \}) \}) \}) \tilde{c} \end{aligned}$$

Figure 4. Recursive medium process for the commit protocol, as in Example 4.3.

Then, associating participants **A**, **B**, and **C** in G_{rc} to names a , b , and c , respectively, process $\mathbf{M}^\mu \llbracket G_{rc} \rrbracket_k$ is as in Figure 4.

We now introduce the third class of mediums, *annotated mediums*. As in Definition 4.2, also in this case it is convenient to consider an additional fresh session k . However, rather than signaling termination/recursion, in this case we use k to emit an observable signal on k for each action of G . To this end, below we use the following notational conventions. First, we write $k.P$ to stand for process $k(x).P$ whenever x is not relevant in P . Also, $\widehat{k}.P$ stands for the process $\bar{k}(v).(\mathbf{0} \mid P)$ in which name v is unimportant.

Definition 4.4 (Annotated Mediums). *Let $G \in \mathcal{G}^\mu$ be a global type. Also, let k be a fresh name. The annotated medium of G with respect to k , denoted $\mathcal{M}^\mu \llbracket G \rrbracket_k$, is defined inductively as follows:*

- $\mathcal{M}^\mu \llbracket \text{end} \rrbracket_k = \mathbf{0}$
- $\mathcal{M}^\mu \llbracket p \rightarrow q : \{ \mathbf{1}_i \langle U_i \rangle . G_i \}_{i \in I} \rrbracket_k =$
 $c_p \triangleright \{ \mathbf{1}_i : k \triangleleft \mathbf{1}_i; c_p(u). \widehat{k}. (c_q \triangleleft \mathbf{1}_i; k \triangleright \{ \mathbf{1}_i : \bar{c}_q(v).([u \leftrightarrow v] \mid k. \mathcal{M}^\mu \llbracket G_i \rrbracket_k) \}_{i \in I}) \} \}_{i \in I}$
- $\mathcal{M}^\mu \llbracket \mu \mathcal{X}. G \rrbracket_k = (\text{corec } \mathcal{X}(\tilde{z}). \mathcal{M}^\mu \llbracket G \rrbracket_k) \tilde{c}$
- $\mathcal{M}^\mu \llbracket \mathcal{X} \rrbracket_k = \mathcal{X}(\tilde{z})$

The key case is $\mathcal{M}^\mu \llbracket p \rightarrow q : \{ \mathbf{1}_i \langle U_i \rangle . G_i \}_{i \in I} \rrbracket_k$. First, the selection of label $\mathbf{1}_i$ by p is followed by a selection of $k \triangleleft \mathbf{1}_i$; then, the output from p , captured by the medium by the input $c_p(u)$, is followed by an output on k ; subsequently, the selection on c_q of label $\mathbf{1}_i$ is followed by a branching in k on label $\mathbf{1}_i$; finally, the output $\bar{c}_q(v)$ is signaled by an input on k , which prefixes the execution of the continuation $\mathcal{M}^\mu \llbracket G_i \rrbracket_k$. This way, actions on k induce a fine-grained correspondence with the behavior of G .

We assume the following name convention for mediums: the actions of every participant p in G are described in $\mathbf{M} \llbracket G \rrbracket$ by prefixes on name c_p (similarly for $\mathbf{M}^\mu \llbracket G \rrbracket_k$ and $\mathcal{M}^\mu \llbracket G \rrbracket_k$). Since in labeled communications $G = p \rightarrow q : \{ \mathbf{1}_i \langle U_i \rangle . G_i \}_{i \in I}$ we always assume $p \neq q$, in $\mathbf{M} \llbracket G \rrbracket$ we will have $c_p \neq c_q$ (similarly for $\mathbf{M}^\mu \llbracket G \rrbracket_k$ and $\mathcal{M}^\mu \llbracket G \rrbracket_k$). Due to this convention we have:

Fact 4.5. *Let G and $\mathbf{M} \llbracket G \rrbracket$ be a global type and its medium, resp. For each $\mathbf{r}_j \in \text{part}(G)$ there is a name $c_j \in \text{fn}(\mathbf{M} \llbracket G \rrbracket)$. (And analogously for $\mathbf{M}^\mu \llbracket G \rrbracket_k$ and $\mathcal{M}^\mu \llbracket G \rrbracket_k$.)*

We stress that from the standpoint of the protocol participants, the existence of the medium is inessential: the local implementations may be constructed (and type-checked) exactly as prescribed by their projected local types, unaware of the medium and its internal structure. In contrast, the medium *depends* on well-behaved participants as stipulated by the global type. In the following we will formalize these intuitions, using the theory of binary session types described in §3.2. We will then be able to formally define the dependence of well-typed mediums on local participants which are well-typed with respect to projections of the given global type.

We first present characterization results for global types in \mathcal{G}^{fin} (§4.2). We extend these results to global types in \mathcal{G}^μ (§4.3).

4.2 Relating Well-Formed Global Types and Typed Mediums: The Finite Case

We formally relate a global type $G \in \mathcal{G}^{\text{fin}}$, its associated medium $\mathbf{M} \llbracket G \rrbracket$, and its corresponding local types $G \upharpoonright p_1, \dots, G \upharpoonright p_n$. We first

introduce some useful auxiliary notions. *Compositional typings* are a class of type judgments which in line with the name convention for mediums (cf. Fact 4.5). Below, we sometimes write $\Gamma; \Delta \vdash_\eta \mathbf{M} \llbracket G \rrbracket$ instead of $\Gamma; \Delta \vdash_\eta \mathbf{M} \llbracket G \rrbracket :: z : \mathbf{1}$, when $z \notin \text{fn}(\mathbf{M} \llbracket G \rrbracket)$.

Definition 4.6 (Compositional Typing). *Let G be a global type. We say that judgement $\Gamma; \Delta \vdash \mathbf{M} \llbracket G \rrbracket :: z : C$ is a compositional typing for $\mathbf{M} \llbracket G \rrbracket$ if: (i) it is a valid typing derivation; (ii) $\Delta = c_1 : A_1, \dots, c_n : A_n$; (iii) for all $\mathbf{r}_i \in G$ there is a $c_i : A_i \in \Delta$; (iv) $C = \mathbf{1}$. In case only conditions (i)–(iii) hold, we say that the judgement is a left-compositional typing for $\mathbf{M} \llbracket G \rrbracket$.*

Intuitively, compositional typings formalize the intuitions hinted at the end of §4.1. These typed interfaces formalize the fact that the medium does not offer any behaviors of its own (cf. the right-hand side $z : \mathbf{1}$) while depending on behaviors which should be available on its free names (cf. the condition on left-hand side typing Δ).

The main difference between local types and binary session types is that the latter do not mention participants. Below, we use B to range over base types (bool, nat, ...) in Definition 3.1.

Definition 4.7 (Local Types and Binary Types). *Mapping $\langle \cdot \rangle$ from local types T (cf. Def. 3.1) into binary types A (cf. Def. 3.6) is inductively defined as:*

$$\begin{aligned} \langle \text{end} \rangle &= \langle B \rangle = \mathbf{1} \\ \langle p! \{ \mathbf{1}_i \langle U_i \rangle . T_i \}_{i \in I} \rangle &= \oplus \{ \mathbf{1}_i : \langle U_i \rangle \otimes \langle T_i \rangle \}_{i \in I} \\ \langle p? \{ \mathbf{1}_i \langle U_i \rangle . T_i \}_{i \in I} \rangle &= \otimes \{ \mathbf{1}_i : \langle U_i \rangle \multimap \langle T_i \rangle \}_{i \in I} \end{aligned}$$

Given a global type G , we now formally relate process $\mathbf{M} \llbracket G \rrbracket$ (typed with a compositional typing) and binary session types representing the projections of G .

4.2.1 Characterization Results

We now present the key correspondence results between global types and well-typed finite mediums (Theorems 4.8 and 4.10). The first direction of the characterization says that well-formedness of global types (Def. 3.4) suffice to ensure compositional typings for mediums with (linear logic based) binary session types:

Theorem 4.8 (From Well-Formedness To Typed Mediums). *Let $G \in \mathcal{G}^{\text{fin}}$ be a global type, with $\text{part}(G) = \{p_1, \dots, p_n\}$. If G is WF (Def. 3.4) then*

$$\Gamma; c_1 : \langle G \upharpoonright p_1 \rangle, \dots, c_n : \langle G \upharpoonright p_n \rangle \vdash \mathbf{M} \llbracket G \rrbracket$$

is a compositional typing for $\mathbf{M} \llbracket G \rrbracket$, for some Γ .

Proof. By a structural induction on G ; see Appendix B.2. In case

$$G = p \rightarrow q : \{ \mathbf{1}_i \langle U_i \rangle . G_i \}_{i \in I} \upharpoonright \mathbf{r}$$

with $\{\mathbf{r}\} \# \{p, q\}$, the flexibility given by \sqcup (Def. 3.2) results into \otimes types in the left-hand side typing for $\mathbf{M} \llbracket G \rrbracket$ which may not be identical. To derive the desired compositional typing, we use rule (T&L₂) so as to silently add/remove labeled options in left \otimes types until achieving identical typings (as required to use rule (T&L)). The case $G = G_1 \mid G_2$ uses independent parallel composition. \square

The following theorem states the converse of Theorem 4.8: it says that compositional typings for mediums induce global types which

$$\begin{array}{c}
\text{(SW1)} \frac{\{p_1, q_1\} \# \{p_2, q_2\}}{p_1 \rightarrow q_1 : \{1_i \langle U_i \rangle . p_2 \rightarrow q_2 : \{1'_j \langle U'_j \rangle . G_{ij}\}_{j \in J}\}_{i \in I} \simeq_{\text{sw}} p_2 \rightarrow q_2 : \{1'_j \langle U'_j \rangle . p_1 \rightarrow q_1 : \{1_i \langle U_i \rangle . G_{ij}\}_{i \in I}\}_{j \in J}} \\
\text{(SW2)} \frac{\{p, q\} \# \text{part}(G_1) \quad \forall i, j \in I. G_i^1 = G_j^1}{p \rightarrow q : \{1_i \langle U_i \rangle . (G_i^1 \mid G_i^2)\}_{i \in I} \simeq_{\text{sw}} G_1^1 \mid p \rightarrow q : \{1_i \langle U_i \rangle . G_i^2\}_{i \in I}}
\end{array}$$

Figure 5. Swapping on global types (cf. Def. 4.12). $A \# B$ denotes that sets A, B are disjoint. The symmetric of (SW2) is omitted.

are WF. We require the following auxiliary definition, which relies on the merge operator given in Definition 3.2.

Definition 4.9. Given local types T_1, T_2 , we write $T_1 \preceq^\sqcup T_2$ if there exists a local type T' such that $T_1 \sqcup T' = T_2$.

Theorem 4.10 (From Well-Typedness To WF Global Types). *Let $G \in \mathcal{G}^{\text{fin}}$ be a global type. If $\Gamma; c_1:A_1, \dots, c_n:A_n \vdash M[G]$ is a compositional typing for $M[G]$ then $\exists T_1, \dots, T_n$ s.t. $G \upharpoonright \mathbf{r}_j \preceq^\sqcup T_j$ and $\langle\langle T_j \rangle\rangle = A_j$, for all $\mathbf{r}_j \in G$.*

The proof of Theorem 4.10 is by structural induction on G ; see Appendix B.3. Observe how notation \preceq^\sqcup allows us to handle the occurrence of labeled alternatives which may be silently introduced by rule (T&L₂).

Remarkably, our results tightly and formally connect global types (in \mathcal{G}^{fin}), local types, and projection (on the multiparty approach) and medium processes and deadlock-free binary session types (rooted in linear logic). Our results provide an independent deep justification, through purely logical arguments, to the forms of projection proposed in the literature. We do not know of works in which the semantics of global type projection is compared/assessed based on different foundations; this also seems an interesting contribution of our approach to multiparty protocol analysis.

Remark 4.11. Theorems 4.8 and 4.10 concern well-formed global types as in [12, 13]. The theorems hold also when global types are well-formed as in [17]; we call those types simply well-formed (or SWF). In the analog of Theorem 4.8 for SWF global types, the proof is simpler as projectibility in [17] ensures identical behavior in all branches. See Appendix B.1.

4.2.2 A Behavioral Characterization of Global Swapping

The *swapping relation* over global types was proposed in [6] as a way of enabling behavior-preserving transformations among causally independent communications. Such transformations may represent optimizations, in which parallelism is increased while preserving the overall intended semantics. We now show a characterization of swapping on global types in terms of a typed behavioral equivalence on mediums.

Definition 4.12 (Swapping for Global Types). *We define swapping, denoted \simeq_{sw} , as the smallest congruence on global types which satisfies the rules in Fig. 5.*

To characterize swapping, we briefly discuss *proof conversions* and *typed behavioral equivalences* for logic-based binary session types, following [25]. The correspondence in [3, 29] is realized by relating *proof conversions* in linear logic with appropriate behavioral equivalences in the process setting. Most conversions correspond to either reductions or structural congruences at the level of processes. There is a group of *commuting conversions* which actually induce a behavioral congruence on typed processes, denoted \simeq_c . Process equalities justified by \simeq_c include, e.g., the following

ones:

$$\begin{aligned}
& (\nu x)(P \mid y(z).Q) \simeq_c y(z).(\nu x)(P \mid Q) \\
& (\nu x)(P \mid \overline{y}(z).(Q \mid R)) \simeq_c \overline{y}(z).(Q \mid (\nu x)(P \mid R)) \\
& (\nu x)(P \mid y \triangleleft 1_i; Q) \simeq_c y \triangleleft 1_i; (\nu x)(P \mid Q)
\end{aligned}$$

Processes equated by \simeq_c are syntactically very different and yet they are associated to the session typed (contextual) behaviour. These equalities reflect a natural typed behavioral equivalence over session-typed processes, called *typed context bisimilarity* [25]. Roughly, typed processes $\Gamma; \Delta \vdash P :: x:A$ and $\Gamma; \Delta \vdash Q :: x:A$ are typed context bisimilar, denoted $\Gamma; \Delta \vdash P \approx Q :: x:A$ if, once composed with their requirements (as described by Γ and Δ), they perform the same actions on x (following A). Typed context bisimilarity is a congruence on well-typed processes.

Theorem 4.13 ([25]). *Let P, Q be well-typed processes. If $\Gamma; \Delta \vdash P \simeq_c Q :: z:C$ then $\Gamma; \Delta \vdash P \approx Q :: z:C$.*

It turns out that swapping in global types (Def. 4.12) can also be directly justified from crisper, more primitive notions, based on the correspondence established by Theorems 4.8 and 4.10. Indeed, by formalizing the behavior of a global type in terms of its medium we may reduce transformations at the level of global types to sound transformations at the level of processes.

Theorem 4.14 below gives a strong connection between swapping on global types (\simeq_{sw}) with typed context bisimilarity (\approx), as motivated above (and defined by Pérez et al. in [25]). Thanks to the theorem, the sequentiality of mediums can be relaxed in the case of causally independent communications formalized by swapping.

Theorem 4.14. *Let $G_1 \in \mathcal{G}^{\text{fin}}$ be a global type, such that $M[G_1]$ has a compositional typing $\Gamma; \Delta \vdash M[G_1]$, for some Γ, Δ . If $G_1 \simeq_{\text{sw}} G_2$ then $\Gamma; \Delta \vdash M[G_1] \approx M[G_2]$.*

Proof (Sketch). The proof proceeds by induction on the definition of \simeq_{sw} (Def. 4.12). To relate swapping with typed context bisimilarity we rely on \simeq_c . We first show that

$$\text{If } G_1 \simeq_{\text{sw}} G_2 \text{ then } \Gamma; \Delta \vdash M[G_1] \simeq_c M[G_2] \quad (3)$$

To establish (3), we exploit the relation between participant identities in global types and names in associated mediums (Fact 4.5): this allows to infer that disjointness conditions for swapping rules imply name distinctions, which in turn enables type-preserving transformations via \simeq_c . The needed transformations rely on equalities detailed by Pérez et al. in [25]. The thesis follows by combining (3) with Theorem 4.13. See Appendix B.4 for details. \square

The converse of Theorem 4.14 does not hold in general: given $M[G]$, the existence of a P' such that $M[G] \simeq_c P'$ does not necessarily imply the existence of a G' such that $G \simeq_{\text{sw}} G'$ and $P' = M[G']$. For instance, consider the global type

$$G_1 = p \rightarrow q : \{1_i \langle U_i \rangle . r \rightarrow p : \{1'_j \langle U'_j \rangle . G_{ij}\}_{j \in J}\}_{i \in I}$$

It cannot be swapped and yet prefixes for q and r in $M[G_1]$ could be commuted. In general, mediums are a fine-grained representation of global types: as a single communication in G is implemented in $M[G]$ using several prefixes, swapping of a type G occurs only when *all* involved prefixes in $M[G]$ can be commuted via \simeq_c . We stress that commutations induced by \simeq_c are always type-preserving; hence, typing for $M[G]$ is invariant under swapping.

4.3 Results for Well-Formed Global Types With Recursion

In this sub-section we consider the language of global types with recursion and without parallel and extend the characterization results in § 4.2. We also present an operational correspondence result.

4.3.1 Characterization Results for Recursive Mediums

We require the following (expected) extension to mapping $\langle\langle\cdot\rangle\rangle$, given in Definition 4.7:

$$\begin{aligned}\langle\langle\mathcal{X}\rangle\rangle &= \mathcal{X} \\ \langle\langle\mu\mathcal{X}.T\rangle\rangle &= \nu\mathcal{X}.\langle\langle T\rangle\rangle\end{aligned}$$

To state the analogous of Theorems 4.8 and 4.10 for co-recursive mediums, we need to close the local projections of a global type. The following definition defines a closure for such recursion variables, using mapping η . Given $\eta = \eta'[\mathcal{X}(\tilde{y}) \mapsto \Gamma; \Delta \vdash k:\mathcal{Y}]$ with $c_i:A_i \in \Delta$ we write $\eta(\mathcal{X})(c_i)$ to denote the type A_i . We write $\text{fv}(G)$ to denote the set of free recursion variables in G .

Definition 4.15 (Closure for Local Types). *Let G , \mathcal{P} , and η be a global type, a set of participants, and a mapping from process variables to typing contexts, respectively. Also, let $\langle\langle\cdot\rangle\rangle$ be the map of Def. 4.7, extended as above. We define:*

$$\bullet \langle\langle G \upharpoonright \mathbf{p}_i \rangle\rangle_{\mathcal{P}}^{\eta} = \begin{cases} \langle\langle G \upharpoonright \mathbf{p}_i \rangle\rangle \{ \eta(\mathcal{X})(c_i) / \mathcal{X} \} & \text{if } \text{fv}(G \upharpoonright \mathbf{p}_i) = \{ \mathcal{X} \} \text{ and } \mathbf{p}_i \in \mathcal{P} \\ \eta(\mathcal{X})(c_i) & \text{if } \text{fv}(G \upharpoonright \mathbf{p}_i) = \{ \mathcal{X} \} \text{ and } \mathbf{p}_i \notin \mathcal{P} \\ \langle\langle G \upharpoonright \mathbf{p}_i \rangle\rangle & \text{if } \text{fv}(G \upharpoonright \mathbf{p}_i) = \emptyset \end{cases}$$

Concerning typing for mediums, the main consequence of adding recursion is that we no longer have $\mathbf{1}$ at the right-hand side typing (cf. Def. 4.6). Intuitively, this is because we can never fully consume a recursive behavior, which is essentially infinite. If recursion is required in the left-hand side typing then some recursive behavior must show up in the right-hand side along name k . (Notice that by Def. 3.3, all the local projections of a recursive global type will be also recursive.)

We now extend Theorems 4.8 and 4.10 to global types in \mathcal{G}^{μ} . As before, the first direction of the characterization says that the conditions that merge-based well-formedness induces on global types suffice to ensure compositional typings for mediums (cf. Definition 4.6):

Theorem 4.16 (From Well-Formedness To Typed Mediums). *Let $G \in \mathcal{G}^{\mu}$ be a global type with $\text{part}(G) = \mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$. If G is WF (Def. 3.4) then*

$$\Gamma; c_1:\langle\langle G \upharpoonright \mathbf{p}_1 \rangle\rangle_{\mathcal{P}}^{\eta}, \dots, c_n:\langle\langle G \upharpoonright \mathbf{p}_n \rangle\rangle_{\mathcal{P}}^{\eta} \vdash_{\eta} \mathbf{M}^{\mu} \llbracket G \rrbracket_k :: k:A$$

is a left compositional typing for $\mathbf{M}^{\mu} \llbracket G \rrbracket_k$ for some Γ, η, A .

We now state the converse of Theorem 4.16. It says that well-typed mediums induce global types which are well-formed. That is, the sequential structure of mediums can be precisely captured by binary session types which have a corresponding local type.

Theorem 4.17 (From Well-Typedness To WF Global Types). *Let $G \in \mathcal{G}^{\mu}$ be a global type, with $\text{part}(G) = \mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$. If*

$$\Gamma; c_1:A_1, \dots, c_n:A_n \vdash_{\eta} \mathbf{M}^{\mu} \llbracket G \rrbracket_k :: k:B$$

is a left compositional typing for $\mathbf{M}^{\mu} \llbracket G \rrbracket_k$ then $\exists T_1, \dots, T_n$ s.t. $G \upharpoonright \mathbf{p}_j \preceq^{\sqcup} T_j$ and $\langle\langle T_j \rangle\rangle_{\mathcal{P}}^{\eta} = A_j$ for all $\mathbf{p}_j \in G$.

4.3.2 Operational Correspondence via Annotated Mediums

The results already presented focus on the static semantics of multiparty and binary systems, and are already key to justify essential properties such as preservation of global deadlock. We now move on to dynamic semantics, and establish the expected precise operational correspondence result between a global type and its medium process (Theorem 4.23). To this end, we rely on the annotated mediums of Definition 4.4: given a global type $G \in \mathcal{G}^{\mu}$, its annotated medium $\mathcal{M}^{\mu} \llbracket G \rrbracket_k$ contains an independent session k which signals the behavior of G . In typing, the observable behavior on k will appear on the right-hand side typing.

The following definition relates the behavior of a global type and that of k .

Definition 4.18 (Global Types and Binary Session Types). *Let $\pi(\cdot)$ denote a mapping from participants to binary session types. The mapping $\langle\langle\cdot\rangle\rangle$ from a global types $G \in \mathcal{G}^{\mu}$ into binary session types A (cf. Def. 3.6) is inductively defined as:*

- $\langle\langle \text{end} \rangle\rangle = \mathbf{1}$
- $\langle\langle \mathbf{p} \rightarrow \mathbf{q} : \{ \mathbf{1}_i : \langle U_i \rangle . G_i \}_{i \in I} \} \rangle = \oplus \{ \mathbf{1}_i : \pi(\mathbf{p}) \otimes \& \{ \mathbf{1}_i : \pi(\mathbf{q}) \multimap \langle G_i \rangle \}_{i \in I} \}$
- $\langle\langle \mathcal{X} \rangle\rangle = \mathcal{X}$
- $\langle\langle \mu\mathcal{X}.G \rangle\rangle = \nu\mathcal{X}.\langle\langle G \rangle\rangle$

For simplicity, we shall assume $\pi(\mathbf{p}) = \mathbf{1}$, for every \mathbf{p} . We may recast Theorem 4.16 above for annotated mediums as follows.

Theorem 4.19 (From Well-Formedness To Typed Annotated Mediums). *Let $G \in \mathcal{G}^{\mu}$ be a global type with $\text{part}(G) = \mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$. If G is WF (Def. 3.4) then judgment*

$$\Gamma; c_1:\langle\langle G \upharpoonright \mathbf{p}_1 \rangle\rangle_{\mathcal{P}}^{\eta}, \dots, c_n:\langle\langle G \upharpoonright \mathbf{p}_n \rangle\rangle_{\mathcal{P}}^{\eta} \vdash_{\eta} \mathcal{M}^{\mu} \llbracket G \rrbracket_k :: k:\langle\langle G \rangle\rangle$$

is well-typed, for some Γ .

The proof of Theorem 4.19 extends the one for Theorem 4.16 by considering session k , which is causally independent from all other sessions of the medium. An analogous of Theorem 4.17 holds for annotated mediums. Because of the silent character of rule $(T \oplus R_2)$, we require some additional notation. Below we write $A_1 \preceq^{\oplus} A_2$ iff either $A_1 = A_2$ or $A_1 = \oplus \{ \mathbf{1}_i : A_i \}_{i \in I}$ and $A_2 = \oplus \{ \mathbf{1}_j : A_j \}_{j \in I \cup J}$, for some J .

Theorem 4.20 (From Well-Typed Annotated Mediums To WF Global Types). *Let $G \in \mathcal{G}^{\mu}$ be a global type. If the following judgment is well-typed*

$$\Gamma; c_1:A_1, \dots, c_n:A_n \vdash \mathcal{M}^{\mu} \llbracket G \rrbracket_k :: k:A_0$$

then $\langle\langle G \rangle\rangle \preceq^{\oplus} A_0$ and $\exists T_1, \dots, T_n$ s.t. $G \upharpoonright \mathbf{r}_j \preceq^{\sqcup} T_j$ and $\langle\langle T_j \rangle\rangle_{\mathcal{P}}^{\eta} = A_j$ for all $\mathbf{r}_j \in G$.

The operational correspondence between global types and annotated mediums is given by Theorem 4.23 below.

To state operational correspondence, we consider the set of *multiparty systems* of a global type. Intuitively, given a global type G , a particular multiparty system is obtained by the composition of well-typed implementations of the local behaviors stipulated by G (with no linear/shared dependencies) with the annotated medium $\mathcal{M}^{\mu} \llbracket G \rrbracket_k$, which provides the “glue code” for connecting them all. We write $\mathcal{S}^k(G)$ to denote the set of all multiparty systems of G ; hence, by construction, any $P \in \mathcal{S}^k(G)$ is a particular implementation of the multiparty conversations specified by G . More formally, we require the following auxiliary definition.

Definition 4.21 (Closure). *Let $\Delta = \{x_j:A_j\}_{j \in J}$ be a linear typing environment. We define the set of processes \mathcal{C}_{Δ} as:*

$$\mathcal{C}_{\Delta} \stackrel{\text{def}}{=} \left\{ \prod_{j \in J} Q_j \mid \cdot; \vdash Q_j :: x_j:A_j \right\}$$

Using closures, we can now define systems:

Definition 4.22 (System). *Let $G \in \mathcal{G}^{\mu}$ be a WF global type, such that $\text{part}(G) = \mathcal{P} = \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$. Also, let*

$$\Delta = c_1:\langle\langle G \upharpoonright \mathbf{p}_1 \rangle\rangle_{\mathcal{P}}^{\eta}, \dots, c_n:\langle\langle G \upharpoonright \mathbf{p}_n \rangle\rangle_{\mathcal{P}}^{\eta}$$

be an environment such that $\Gamma; \Delta \vdash \mathcal{M}^{\mu} \llbracket G \rrbracket_k :: k:\langle\langle G \rangle\rangle$, for some Γ . The set of multiparty systems of G , written $\mathcal{S}^k(G)$, is defined as:

$$\{ (\nu c_{\mathbf{p}_1}, \dots, \nu c_{\mathbf{p}_n})(Q \mid \mathcal{M}^{\mu} \llbracket G \rrbracket_k) \mid Q \in \mathcal{C}_{\Delta} \}$$

By construction, processes in $\mathcal{S}^k(G)$ will only have observable behavior on name k . We rely on labeled transition systems (LTSs) for processes and global types; they are denoted $P \xrightarrow{\lambda} P'$ and $G \xrightarrow{\sigma} G'$, respectively. While the former is standard for session π -calculi (see, e.g., [3]), the latter results by extending the LTS in [12] with intermediate states. Given a name k and a participant p , two auxiliary mappings, denoted $\llbracket \cdot \rrbracket^k$ and $\llbracket \cdot \rrbracket_p$, tightly relate global type labels σ to process labels λ . (See Appendix A.4 for details in the LTS for global types and the auxiliary mappings.) We have:

Theorem 4.23 (Global Types and Mediums: Operational Correspondence). *Let $G \in \mathcal{G}^\mu$ be a WF global type and P any process in $\mathcal{S}^k(G)$. We have:*

- (a) *If $G \xrightarrow{\sigma} G'$ then there exist λ, P' s.t. $P \xRightarrow{\lambda} P'$, $\lambda = \llbracket \sigma \rrbracket^k$, and $P' \in \mathcal{S}^k(G')$.*
- (b) *If there is some P_0 s.t. $P \Rightarrow P_0 \xrightarrow{\lambda} P'$ with $\lambda \neq \tau$ then there exist σ, G' s.t. $G \xrightarrow{\sigma} G'$, $\text{subj}(\sigma) = p$, $\sigma = \llbracket \lambda \rrbracket_p$, and $P' \in \mathcal{S}^k(G')$.*

By means of this operational correspondence between a global type G and its process implementations (as captured by set $\mathcal{S}^k(G)$), we confirm that (annotated) medium processes faithfully mirror the communicating behavior of the given global type.

4.4 The Tension Between Global Types for Composition and Recursion

Distinguishing between \mathcal{G}^{fin} and \mathcal{G}^μ has a conceptual justification, as discussed in § 4.1. There is also a more technical motivation for this distinction, related to typability. The reason why finite mediums (Def. 4.1) can support the composition of global types (as originally proposed in [17]) is the following: as finite mediums do not have behavior on their own (i.e., compositional typings ensure that their right-hand side typing is 1) they are amenable to independent parallel composition, as supported by binary session types. This kind of parallel composition neatly coincides with well-formedness for composition of global types in [17], which allow composition the global types with disjoint senders and receivers (cf. Def. 3.3). We find it remarkable that finite mediums are able to cleanly justify natural requirements for multiparty session types.

Independent parallel composition is no longer possible when we move to co-inductive types, which are needed to type the mediums for global types in \mathcal{G}^μ (cf. Defs. 4.2 and 4.4). In fact, when the right-hand side typing is different from 1 we are not able to type the composition of independent processes. Nevertheless, slightly less general forms of composition of global types are still possible. For instance, in Def. 4.4, we could have combined non-annotated and annotated mediums as in, e.g., $M[G_i] \mid M^\mu[G_j]_k$. This resulting “hybrid annotated medium” is typable, with type $k: \langle G_j \rangle$ in the right-hand side. We find these forms of composition useful for modular reasoning on multiparty systems.

5. Sharing in Finite Multiparty Conversations

Here we further illustrate reasoning about global types in \mathcal{G}^{fin} exploiting the properties given in § 4.2. In particular, we show that the absence of recursive types does not necessarily preclude specifying and reasoning about non-trivial forms of replication and sharing.

As an example, let us consider a variant of the two-buyer protocol in [17], in which two buyers (B_1 and B_2) coordinate to buy an item from a seller (S). The three-party interaction is given by the following global type:

$$G_{BS} = B_1 \rightarrow S : \{ \text{send} \langle \text{str} \rangle . S \rightarrow B_1 : \{ \text{rep} \langle \text{int} \rangle . S \rightarrow B_2 : \{ \text{rep} \langle \text{int} \rangle . B_1 \rightarrow B_2 : \{ \text{shr} \langle \text{int} \rangle . B_2 \rightarrow S : \{ \text{ok} \langle 1 \rangle . \text{end} , \text{quit} \langle 1 \rangle . \text{end} \} \} \} \}$$

We omit the (easy) definition of process $M[G_{BS}]$, and proceed to examine its properties. Relying on Theorems 4.8 and 4.10, we have the compositional typing:

$$\Gamma ; c_1 : B_1, c_2 : S, c_3 : B_2 \vdash_\eta M[G_{BS}] :: - : 1 \quad (4)$$

for some Γ and with $B_1 = \langle G_{BS} \upharpoonright B_1 \rangle$, $S = \langle G_{BS} \upharpoonright S \rangle$, and $B_2 = \langle G_{BS} \upharpoonright B_2 \rangle$. To implement the protocol, one may simply compose $M[G_{BS}]$ with type compatible processes $\cdot ; \vdash \text{Buy}1 :: c_1 : B_1$, $\cdot ; \vdash \text{Sel} :: c_2 : S$, and $\cdot ; \vdash \text{Buy}2 :: c_3 : B_2$:

$$\Gamma ; \cdot \vdash_\eta (\nu c_1)(\text{Buy}1 \mid (\nu c_2)(\text{Sel} \mid (\nu c_3)(\text{Buy}2 \mid M[G_{BS}]))) \quad (5)$$

The binary session types in § 3.2 allows us to infer that the multiparty system defined by (5) adheres to the declared projected types, is lock-free, and non-diverging. Just as we inherit strong properties for *Buy1*, *Sel*, and *Buy2* above, we may inherit the same properties for more interesting system configurations. In particular, local implementations which appeal to replication and sharing, admit also precise analyses thanks to the characterizations in § 4.2. Let us consider a setting in which the processes to be composed with the medium must be invoked from a replicated service (a source of generic process definitions). We may have:

$$\begin{aligned} \cdot ; \vdash_\eta !u_1(w). \text{Buy}1_w :: u_1 : !B_1 & \quad \cdot ; \vdash_\eta !u_2(w). \text{Sel}_w :: u_2 : !S \\ \cdot ; \vdash_\eta !u_3(w). \text{Buy}2_w :: u_3 : !B_2 \end{aligned}$$

and the following “initiator processes” would spawn a copy of the medium’s requirements, instantiated at appropriate names:

$$\begin{aligned} \cdot ; u_1 : !B_1 \vdash_\eta \overline{u_1}(x).[x \leftrightarrow c_1] :: c_1 : B_1 \\ \cdot ; u_2 : !S \vdash_\eta \overline{u_2}(x).[x \leftrightarrow c_2] :: c_2 : S \\ \cdot ; u_3 : !B_2 \vdash_\eta \overline{u_3}(x).[x \leftrightarrow c_3] :: c_3 : B_2 \end{aligned}$$

Let us write *RBuy1*, *RBuy2*, and *RSel* to denote the composition of replicated definitions and initiators above. Intuitively, they represent the “remote” variants of *Buy1*, *Buy2*, and *RSel*, respectively. We may then define the multiparty system:

$$\Gamma ; \cdot \vdash_\eta (\nu c_1)(R\text{Buy}1 \mid (\nu c_2)(R\text{Sel} \mid (\nu c_3)(R\text{Buy}2 \mid M[G_{BS}])))$$

which, with a concise specification, improves (5) with concurrent invocation/instantiation of replicated service definitions. As (5), the revised composition above is correct, lock-free, and terminating.

Rather than appealing to initiators, a scheme in which the medium invokes and instantiates services directly is also expressible in our framework, in a type consistent way. Using (4), and assuming $\Gamma = u_1 : B_1, u_2 : S, u_3 : B_2$, we may derive:

$$\Gamma ; \cdot \vdash_\eta \overline{u_1}(c_1).\overline{u_2}(c_2).\overline{u_3}(c_3).M[G_{BS}] \quad (6)$$

Hence, prior to engage in the mediation behavior for G_{BS} , the medium first spawns a copy of the required services. We may relate the guarded process in (6) with the multicast session request construct in multiparty session processes [17]. Observe that (6) cleanly distinguishes between session initiation and actual communication behavior: the distinction is given at the level of processes (cf. output prefixes on u_1 , u_2 , and u_3) but also at the level of typed interfaces.

The service invocation (6) may be regarded as “eager”: all required services must be sequentially invoked prior to executing the protocol. We may also obtain, in a type-consistent manner, a medium process implementing a “lazy” invocation strategy that spawns services only when necessary. For the sake of example, consider process $Eager_{BS} \triangleq \overline{u_3}(c_3).M[G_{BS}]$ in which only the invocation on u_3 is blocking the protocol, with “open” dependencies on c_1, c_2 . That is, we have $\Gamma ; c_1 : B_1, c_2 : S \vdash Eager_{BS} :: z : 1$. It could be desirable to postpone the invocation on u_3 as much as possible. By combining the commutations on process prefixes realised by \simeq_c [25] and Theorem 4.13, we may obtain:

$$\Gamma ; c_1 : B_1, c_2 : S \vdash Eager_{BS} \approx Lazy_{BS} :: - : 1$$

where $Lazy_{BS}$ is the process obtained from $Eager_{BS}$ by “pushing inside” prefix $\overline{u}_3(c_3)$ as deep as possible in the process structure.

6. Further Developments and Extensions

We now briefly describe two possible extensions to multiparty session types which exploit our analysis technique based on mediums.

Adding A Join Primitive. As we have seen, the mediums offer a clean and simple representation for name-passing in multiparty exchanges. Exploiting this feature, we may extend the syntax of global types with a primitive join $s(\mathbf{r}).G$, which denotes the fact that participant \mathbf{r} , declared in G , is to be realized by invoking a shared service s . This kind of primitive can be found in Conversation Types [4], but has not been yet considered within multiparty session types. Let $G_1(\mathbf{r})$ and $G_2(\mathbf{r})$ be two global types in which participant \mathbf{r} is declared. We may write, e.g., the global type

$$p \rightarrow q: \{ \mathbf{1}_1 \langle \text{int} \rangle . \text{join } s_1(\mathbf{r}).G_1(\mathbf{r}), \mathbf{1}_2 \langle \text{bool} \rangle . \text{join } s_2(\mathbf{r}).G_2(\mathbf{r}) \}$$

in which participant \mathbf{r} may be implemented by different shared services (s_1 or s_2) depending on the selected label. The medium for this primitive would be:

$$M[\text{join } s(\mathbf{r}).G] = \overline{s}(c_{\mathbf{r}}).M[G]$$

Suppose that $\llbracket G(\mathbf{r}) \rrbracket \mathbf{r} \rrbracket = A_{\mathbf{r}}$. The above medium could be typed in the system of [3, 29] as follows:

$$\Gamma; \Delta, s : !A_{\mathbf{r}}, \vdash_{\mathbf{r}} \overline{s}(c_{\mathbf{r}}).M[G] :: z:1$$

where Δ describes the behaviors of other participants in G , reflecting the fact that s is a shared service.

Parametric Polymorphism. Building upon mediums, we may also extend known multiparty session type theories with features well-understood in the binary setting but not yet developed for multiparty sessions. A particularly relevant such features is *parametric polymorphism* (in the style of the Girard-Reynolds polymorphic λ -calculus), studied for binary sessions by Caires et al. [5] and Wadler [31]. We do not know of multiparty session theories supporting polymorphism; so an extension through our approach would be particularly significant.

We follow the approach in [5], which extends the system of [3] with two kinds of session types, $\forall X.A$ and $\exists X.A$, corresponding to impredicative universal and existential quantification over sessions. They are interpreted as the input and output of a session type, respectively. The syntax of processes is extended accordingly, with prefixes $\overline{x}A.P$ and $x(X).P$. To define global types with polymorphism, we may extend the syntax of U in Def. 3.1 with session types A . Here is a simple example of a polymorphic global type:

$$G_{\text{poly}} = p \rightarrow q: \{ \mathbf{1}_1 \langle A \rangle . G \}$$

Global type G_{poly} abstracts a scenario in which p sends to q a session type A using label $\mathbf{1}_1$. This means that the local implementation for q should be *parametric* on any session type which is to be received from p . We would have the following medium for G_{poly} :

$$M[p \rightarrow q: \{ \mathbf{1}_1 \langle A \rangle . G \}] = c_p \triangleright \{ \mathbf{1}_1 : c_p(X).c_q \triangleleft \mathbf{1}_1; \overline{c_q} X.M[G_i] \}$$

It is worth stressing that this extension should be completely orthogonal to the results in § 4.2, for the polymorphic binary sessions in [5] are type-preserving, deadlock-free, and terminating.

7. Related Work

As already discussed, the key obstacle in reducing multiparty session types into binary ones consists in defining binary fragments which preserve the sequencing information of the global specification. *Correspondence assertions* [2] offer one way of relating otherwise independent binary sessions. Present in the syntax of processes and types, such assertions may track data dependencies and

detect unintended operations. Retaining a standard syntax for binary and multiparty types, here we capture the sequencing information using a process extracted from a global type. Our approach relies on deadlock-freedom (not available in [2]) and offers a principled way of transferring it to multiparty systems. In [26, 28] Toninho et al. studied the integration of assertions in session types via dependent types and authorization logics, allowing expressive certified contracts; based on the results in this paper, the added expressiveness brought in by [26, 28] would carry to the multiparty setting, similarly as described for parametric polymorphism in § 6.

Typed frameworks for multiparty interactions were first proposed in [1, 17]. To our knowledge, ours is the first formal characterization of multiparty session types using binary session types. Previous works have encoded binary session types into other type systems. For instance, [10] encodes binary session types into the linear types of [19]. Combined with [10], our work connects a standard theory of global types with the linear types of [19]; this further results appears new, and deserves investigation. Related to this, as a case study for a theory of deadlock-free processes, the work [24] identified a class of multiparty systems for which the analysis of deadlock-freedom can be reduced to analysis of linear π -calculus processes. In contrast with our work, the reduction in [24] does not establish formal connections with binary session types, nor exploits other properties of processes to reason about global specifications.

Building upon [3], in [23, Ch. 4] a correspondence between two-party choreographies and proofs from LCL, a linear logic with hypersequents, is given. Projection is cleanly defined at the level of proofs, but the analysis of n -ary choreographies, exponentials, and forms of iterative behavior are left for future work.

Our medium processes, the key technical device in our developments, are loosely related to the concept of *orchestrators* in service-oriented programming. The work [21] shows how to synthesize an orchestrator from a service choreography, using finite state machines to model both choreography and orchestrator, which already distinguishes this work from ours. We consider choreographies specified as behavioral types; mediums are processes obtained directly from those types. Our work has a foundational character, for it formally connects communicating automata (related to global types) and a Curry-Howard correspondence based on linear logic propositions (which supports binary session types); in contrast, the results in [21] have a more pragmatic spirit, for the goal is to generate Petri net and BPMN models from the obtained orchestrator.

8. Concluding Remarks

We have developed a comprehensive analysis of multiparty session types on top of an elementary type theory for binary sessions. Our results rely on *medium processes*, a simple but effective characterization of multiparty interactions as expressed by standard global types. Using well-typed mediums under the theory of (linear logic based) binary session types in [3, 29] we obtained strong characterizations of mediums with respect to the projections of a global type. Such characterizations allow us to uniformly transfer to the multiparty setting key properties of the binary session theory (notably, deadlock-freedom and behavioral equivalences). In our view, our characterizations do not diminish the relevance of existing frameworks of multiparty sessions. Rather, it is most reasonable that in applications the analysis of multiparty protocols can be effectively done with frameworks in which multiparty interaction is a first-class idiom. On the other hand, our results provide further evidence of the fundamental character of key ingredients in multiparty session types, and build on (perhaps unexpected, but certainly welcome) tight connections between two independently motivated theories of session types with foundational significance: one based on linear logic [3], the other based on communicating automata [12]. These correspondences should be further explored in future re-

search, in connection with more expressive types (e.g., dependent types) and computational models (e.g. asynchrony).

References

- [1] E. Bonelli and A. B. Compagnoni. Multipoint session types for a distributed calculus. In *TGC*. Springer, 2007.
- [2] E. Bonelli, A. Compagnoni, and E. Gunter. Correspondence assertions for process synchronization in concurrent communications. *J. Funct. Program.*, 15:219–247, 2005. ISSN 0956-7968. .
- [3] L. Caires and F. Pfenning. Session types as intuitionistic linear propositions. In *CONCUR’2010*, LNCS. Springer, 2010.
- [4] L. Caires and H. T. Vieira. Conversation types. *Theor. Comput. Sci.*, 411(51-52):4399–4440, 2010.
- [5] L. Caires, J. A. Pérez, F. Pfenning, and B. Toninho. Behavioral polymorphism and parametricity in session-based communication. In *ESOP*, volume 7792 of LNCS, pages 330–349. Springer, 2013.
- [6] M. Carbone and F. Montesi. Deadlock-freedom-by-design: multiparty asynchronous global programming. In *POPL*, pages 263–274. ACM, 2013. ISBN 978-1-4503-1832-7.
- [7] G. Castagna, N. Gesbert, and L. Padovani. A Theory of Contracts for Web Services. In *POPL*, ACM SIGPLAN Notices 43, pages 261–272. ACM, 2008. .
- [8] G. Castagna, M. Dezani-Ciancaglini, and L. Padovani. On Global Types and Multi-Party Sessions. *Log. Meth. in Comp. Sci.*, 8:1–45, 2012. .
- [9] M. Coppo, M. Dezani-Ciancaglini, L. Padovani, and N. Yoshida. Inference of global progress properties for dynamically interleaved multiparty sessions. In *Proc. of COORDINATION*. Springer, 2013.
- [10] O. Dardha, E. Giachino, and D. Sangiorgi. Session types revisited. In *PPDP*. ACM, 2012. ISBN 978-1-4503-1522-7.
- [11] P.-M. Deniérou and N. Yoshida. Dynamic multirole session types. In *POPL*, pages 435–446. ACM, 2011.
- [12] P.-M. Deniérou and N. Yoshida. Multiparty compatibility in communicating automata: Characterisation and synthesis of global session types. In *ICALP’13*. Springer, 2013.
- [13] P.-M. Deniérou, N. Yoshida, A. Bejleri, and R. Hu. Parameterised multiparty session types. *Log. Meth. in Comp. Sci.*, 8(4), 2012.
- [14] B. Genest and A. Muscholl. Message sequence charts: A survey. In *ACSD*, pages 2–4. IEEE Computer Society, 2005.
- [15] K. Honda. Types for dyadic interaction. In *CONCUR*, volume 715 of LNCS, pages 509–523. Springer, 1993.
- [16] K. Honda, V. T. Vasconcelos, and M. Kubo. Language primitives and type discipline for structured communication-based programming. In *ESOP’98*, LNCS. Springer, 1998.
- [17] K. Honda, N. Yoshida, and M. Carbone. Multiparty asynchronous session types. In *POPL*, pages 273–284. ACM, 2008.
- [18] A. Igarashi and N. Kobayashi. A generic type system for the pi-calculus. *Theor. Comput. Sci.*, 311(1-3):121–163, 2004.
- [19] N. Kobayashi, B. C. Pierce, and D. N. Turner. Linearity and the pi-calculus. In *POPL*, 1996.
- [20] J. A. McCarthy and S. Krishnamurthi. Cryptographic protocol explication and end-point projection. In *ESORICS*, volume 5283 of LNCS, pages 533–547. Springer, 2008.
- [21] S. McIlvenna, M. Dumas, and M. T. Wynn. Synthesis of orchestrators from service choreographies. In *APCCM*, volume 96 of *CRPIT*, pages 129–138. Australian Computer Society, 2009.
- [22] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, part I/II. *Inf. Comput.*, 100(1):1–77, 1992.
- [23] F. Montesi. *Choreographic Programming*. PhD thesis, IT Univ. of Copenhagen, 2013.
- [24] L. Padovani. Deadlock and lock freedom in the linear π -calculus, 2014. To appear in CSL-LICS’14.
- [25] J. A. Pérez, L. Caires, F. Pfenning, and B. Toninho. Linear logical relations for session-based concurrency. In *ESOP*. Springer, 2012.
- [26] F. Pfenning, L. Caires, and B. Toninho. Proof-carrying code in a session-typed process calculus. In *Proc. of CPP ’11*, volume 7086 of LNCS. Springer, 2011.
- [27] D. Sangiorgi and D. Walker. *The π -calculus: A Theory of Mobile Processes*. Cambridge University Press, 2001.
- [28] B. Toninho, L. Caires, and F. Pfenning. Dependent session types via intuitionistic linear type theory. In *Proc. of PPDP ’11*. ACM, 2011.
- [29] B. Toninho, L. Caires, and F. Pfenning. Corecursion and Non-Divergence in Session Types. In *TGC*, 2014.
- [30] V. T. Vasconcelos. Sessions, from types to programming languages. *Bulletin of the EATCS*, 103:53–73, 2011.
- [31] P. Wadler. Propositions as sessions. *J. Funct. Program.*, 24(2-3):384–418, 2014.

Contents

1	Introduction	1
2	Our Approach and Main Results, By Example	2
3	Preliminaries: Multiparty and Binary Sessions	4
3.1	Multiparty Session Types	4
3.2	Binary Session Types Based on Linear Logic	4
4	Relating Multiparty Protocols and Binary Session Typed Processes	6
4.1	Medium Processes	6
4.2	Relating Well-Formed Global Types and Typed Mediums: The Finite Case	7
4.2.1	Characterization Results	7
4.2.2	A Behavioral Characterization of Global Swapping	8
4.3	Results for Well-Formed Global Types With Recursion	8
4.3.1	Characterization Results for Recursive Mediums	9
4.3.2	Operational Correspondence via Annotated Mediums	9
4.4	The Tension Between Global Types for Composition and Recursion	10
5	Sharing in Finite Multiparty Conversations	10
6	Further Developments and Extensions	11
7	Related Work	11
8	Concluding Remarks	11
A	Additional Definitions for § 4	13
A.1	Independent Composition	13
A.2	Simple Projectability and Well-Formedness	13
A.3	Proof Conversions	14
A.4	Labeled Transition Systems for Processes and Global Types	14
B	Omitted Proofs from § 4.2	16
B.1	Relating SWF Global Types and Typed Mediums	16
B.1.1	SWF Global Types Ensure Compositional Typings	16
B.1.2	Compositional Typings Induce SWF Global Types	18
B.2	Proof of Theorem 4.8: MWF Global Types Ensure Compositional Typings	19
B.3	Proof of Theorem 4.10: Compositional Typings Induce MWF Global Types	20
B.4	Proof of Theorem 4.14: Behavioral Characterization of Swapping	22
C	Omitted Proofs from § 4.3	23
C.1	Characterization Results for Recursive Mediums	23
C.2	Characterization Results for Annotated Mediums	25
C.3	Operational Correspondence via Annotated Mediums	25

A. Additional Definitions for § 4

A.1 Independent Composition

As mentioned in § 3.2, the following rule for *independent parallel composition* is derivable:

$$(\text{INDCOMP}) \frac{\Gamma; \Delta_1 \vdash P :: - : 1 \quad \Gamma; \Delta_2 \vdash Q :: z : C}{\Gamma; \Delta_1, \Delta_2 \vdash P \mid Q :: z : C}$$

where ‘ $-$ ’ denotes a “dummy name” not in $\text{fn}(P)$.

A.2 Simple Projectability and Well-Formedness

Definition A.1 (Simple Projection [17]). *Let G be a global type. The simple projection of G under participant \mathbf{r} , denoted $G \downarrow \mathbf{r}$, is inductively defined as follows:*

- $\text{end} \downarrow \mathbf{r} = \text{end}$
- $\mathbf{p} \twoheadrightarrow \mathbf{q} : \{ \mathbf{l}_i \langle U_i \rangle . G_i \} \downarrow \mathbf{r} = \begin{cases} \mathbf{p} ! \{ \mathbf{l}_i \langle U_i \rangle . G_i \} \downarrow \mathbf{r} & \text{if } \mathbf{r} = \mathbf{p} \\ \mathbf{p} ? \{ \mathbf{l}_i \langle U_i \rangle . G_i \} \downarrow \mathbf{r} & \text{if } \mathbf{r} = \mathbf{q} \\ G_1 \downarrow \mathbf{r} & \text{if } \mathbf{r} \neq \mathbf{p} \text{ and } \mathbf{r} \neq \mathbf{q} \text{ and } \forall i, j \in I. G_i \downarrow \mathbf{r} = G_j \downarrow \mathbf{r} \end{cases}$

$$\begin{aligned}
\Gamma; \Delta, y:A \otimes B \vdash (\nu x)(P \mid y(z).Q) &\simeq_c y(z).(\nu x)(P \mid Q) :: k:E & (I-3) \\
\Gamma; \Delta, y:A \otimes B \vdash (\nu x)(y(z).P \mid Q) &\simeq_c y(z).(\nu x)(P \mid Q) :: k:E & (I-4) \\
\Gamma; \Delta, y:A \multimap B \vdash (\nu x)(P \mid \overline{y}(z).(Q \mid R)) &\simeq_c \overline{y}(z).((\nu x)(P \mid Q) \mid R) :: k:E & (I-6) \\
\Gamma; \Delta, y:A \multimap B \vdash (\nu x)(P \mid \overline{y}(z).(Q \mid R)) &\simeq_c \overline{y}(z).(Q \mid (\nu x)(P \mid R)) :: k:E & (I-7) \\
\Gamma; \Delta, y:A \multimap B \vdash (\nu x)(\overline{y}(z).(Q \mid P) \mid R) &\simeq_c \overline{y}(z).(Q \mid (\nu x)(P \mid R)) :: k:E & (I-8) \\
\Gamma; \Delta, y:A \& B \vdash (\nu x)(P \mid y \triangleleft 1_i; Q) &\simeq_c y \triangleleft 1_i; (\nu x)(P \mid Q) :: k:E & (I-10) \\
\Gamma; \Delta, y:A \oplus B \vdash (\nu x)(P \mid y \triangleright \{Q, R\}) &\simeq_c y \triangleright \{(\nu x)(P \mid Q), (\nu x)(P \mid R)\} :: k:E & (I-14) \\
\Gamma, u:A; \Delta \vdash (\nu x)(P \mid \overline{u}(y).Q) &\simeq_c \overline{u}(y).(\nu x)(P \mid Q) :: k:E & (I-15) \\
\Gamma; \Delta, y:A \& B \vdash (\nu x)(y \triangleleft 1_i; P \mid R) &\simeq_c y \triangleleft 1_i; (\nu x)(P \mid R) :: k:E & (I-18) \\
\Gamma; \Delta, y:A \oplus B \vdash (\nu x)(y \triangleright \{P, Q\} \mid R) &\simeq_c y \triangleright \{(\nu x)(P \mid R), (\nu x)(Q \mid R)\} :: k:E & (I-20) \\
\Gamma; \Delta \vdash (\nu x)(P \{y/u\} \mid Q) &\simeq_c (\nu x)(P \mid Q) \{y/u\} :: k:E & (I-21) \\
\Gamma; \Delta \vdash (\nu x)(P \mid Q \{y/u\}) &\simeq_c (\nu x)(P \mid Q) \{y/u\} :: k:E & (I-22) \\
\Gamma, u:A; \Delta \vdash (\nu x)(\overline{u}(y).P \mid R) &\simeq_c \overline{u}(y).(\nu x)(P \mid R) :: k:E & (I-23) \\
\Gamma, u:A; \Delta \vdash (\nu x)(P \mid \overline{u}(y).R) &\simeq_c \overline{u}(y).(\nu x)(P \mid R) :: k:E & (I-24) \\
\Gamma; \cdot \vdash (\nu u)((!u(y).P) \mid \mathbf{0}) &\simeq_c \mathbf{0} :: -:\mathbf{1} & (I-25) \\
\Gamma; \Delta, y:A \otimes B \vdash (\nu u)((!u(y).P) \mid y(z).Q) &\simeq_c y(z).(\nu u)((!u(y).P) \mid Q) :: k:E & (I-28) \\
\Gamma; \Delta, y:A \multimap B \vdash (\nu u)((!u(y).P) \mid \overline{y}(z).(Q \mid R)) &\simeq_c \overline{y}(z).(((\nu u)(!u(y).P \mid Q) \mid (\nu u)((!u(y).P) \mid R))) :: k:E & (I-30) \\
\Gamma; \Delta, y:A \& B \vdash (\nu u)(!u(z).P \mid y \triangleleft 1_i; Q) &\simeq_c y \triangleleft 1_i; (\nu u)(!u(z).P \mid Q) :: k:E & (I-32) \\
\Gamma; \Delta, y:A \oplus B \vdash (\nu u)(!u(z).P \mid y \triangleright \{Q, R\}) &\simeq_c y \triangleright \{(\nu u)(!u(z).P \mid Q), (\nu u)(!u(z).P \mid R)\} :: k:E & (I-36) \\
\Gamma; \Delta \vdash (\nu u)(!u(y).P \mid Q \{y/v\}) &\simeq_c (\nu u)(!u(y).P \mid Q) \{y/v\} :: k:E & (I-38) \\
\Gamma; \Delta \vdash (\nu u)(!u(y).P \mid \overline{v}(y).Q) &\simeq_c \overline{v}(y).(\nu u)(!u(y).P \mid Q) :: k:E & (I-39)
\end{aligned}$$

Figure 6. Process equalities induced by proof conversions (Part I - cf. Def. A.4)

$$\bullet (G_1 \mid G_2) \wr \mathbf{r} = \begin{cases} G_i \wr \mathbf{r} & \text{if } \mathbf{r} \in G_i \text{ and } \mathbf{r} \notin G_j, \text{ with } i \neq j \in \{1, 2\} \\ \text{end} & \text{if } \mathbf{r} \notin G_1 \text{ and } \mathbf{r} \notin G_2 \end{cases}$$

When a side condition does not hold, the map is undefined.

We then may define *simple well-formedness*:

Definition A.2 (Simple Well-Formedness). *We say that global type G is simply well-formed (SWF, in the following) if for all $\mathbf{r} \in G$, the simple projection $G \wr \mathbf{r}$ is defined.*

In what follows, we often say that a global type is MWF (merge-based well-formed) if it is well-formed according to Def. 3.4.

A.3 Proof Conversions

Figs. 6 and 7 give the commuting conversions relevant to the present development, in particular for Theorem 4.14 (proven in § B.4). Intuitively, they concern the interaction of (i) two left rules and (ii) a left rule and a rule for composition. There are other commuting conversions (see, e.g., [25]); however, given our focus on compositional typings (in which the only typing admitted in the right-hand side typing is $\mathbf{1}$, cf. Def. 4.6) the conversions in Figs. 6 and 7 are the only relevant ones. In the figures, we sometimes appeal to the following notational abbreviations:

Convention A.3 (Additives). *We abbreviate $\mathcal{E}\{1_i:A_i\}_{i \in I}$ as $\mathcal{E}\{1_i:A_i, 1_j:A_j\}$ when $I = \{2\}$. When labels are unimportant, we write $A_i \& A_j$, with labels having left/right readings, as in [3]. Similar abbreviations apply for $\oplus\{1_i:A_i\}_{i \in I}$.*

We may define:

Definition A.4 (Proof Conversions). *We define \simeq_c as the least congruence on processes induced by the process equalities in Figures 6 and 7 (Pages 14–15).*

A.4 Labeled Transition Systems for Processes and Global Types

We now present auxiliary notions, needed for the operational correspondence result stated in § 4.3.2 (and proved in § C.7).

LTS for Processes. To characterize the interactions of a well-typed process with its environment, we extend the early labeled transition system (LTS) for the π -calculus [27] with labels and transition rules for choice and forwarding constructs. A transition $P \xrightarrow{\lambda} Q$ denotes that P may evolve to Q by performing the action represented by label λ . Transition labels are defined below:

$$\lambda ::= \tau \mid x(y) \mid x \triangleleft 1 \mid \overline{x}y \mid \overline{x}(y) \mid \overline{x} \triangleleft 1$$

$\Gamma; \Delta, x:A \otimes B, z:C \otimes D \vdash x(y).z(w).P \simeq_c z(w).x(y).P :: k:E$	(II-1)
$\Gamma; \Delta, z:D \multimap C, x:A \multimap B \vdash \overline{x}(w).(R \mid \overline{x}(y).(P \mid Q)) \simeq_c \overline{x}(y).(P \mid \overline{x}(w).(R \mid Q)) :: k:E$	(II-2)
$\Gamma; \Delta, z:D \multimap C, x:A \multimap B \vdash \overline{x}(w).(R \mid \overline{x}(y).(P \mid Q)) \simeq_c \overline{x}(y).(\overline{x}(w).(R \mid P) \mid Q) :: k:E$	(II-3)
$\Gamma; \Delta, w:C \multimap D, x:A \otimes B \vdash \overline{w}(z).(Q \mid x(y).P) \simeq_c x(y).\overline{w}(z).(Q \mid P) :: k:E$	(II-4)
$\Gamma; \Delta, w:C \multimap D, x:A \otimes B \vdash \overline{w}(z).(x(y).P \mid Q) \simeq_c x(y).\overline{w}(z).(P \mid Q) :: k:E$	(II-5)
$\Gamma, u:A, v:C; \Delta \vdash \overline{u}(y).\overline{v}(x).P \simeq_c \overline{v}(x).\overline{u}(y).P :: k:E$	(II-6)
$\Gamma, u:C; \Delta, x:A \multimap B \vdash \overline{u}(z).\overline{x}(y).(P \mid Q) \simeq_c \overline{x}(y).(\overline{u}(z).P \mid Q) :: k:E$	(II-7)
$\Gamma, u:C; \Delta, x:A \multimap B \vdash \overline{u}(z).\overline{x}(y).(P \mid Q) \simeq_c \overline{x}(y).(P \mid \overline{u}(z).Q) :: k:E$	(II-8)
$\Gamma, u:A; \Delta, z:C \otimes D \vdash \overline{u}(y).z(w).P \simeq_c z(w).\overline{u}(y).P :: k:E$	(II-9)
$\Gamma; \Delta, x:A \oplus B, y:C \oplus D \vdash y \triangleright \{x \triangleright \{P_1, Q_1\}, x \triangleright \{P_2, Q_2\}\} \simeq_c x \triangleright \{y \triangleright \{P_1, P_2\}, y \triangleright \{Q_1, Q_2\}\} :: k:E$	(II-10)
$\Gamma, u:C; \Delta, x:A \oplus B \vdash \overline{u}(z).x \triangleright \{P, Q\} \simeq_c x \triangleright \{\overline{u}(z).P, \overline{u}(z).Q\} :: k:E$	(II-11)
$\Gamma; \Delta, w:A \multimap E, z:C \oplus D \vdash z \triangleright \{\overline{w}(y).(P \mid R_1), \overline{w}(y).(P \mid R_2)\} \simeq_c \overline{w}(y).(P \mid z \triangleright \{R_1, R_2\}) :: k:E$	(II-12)
$\Gamma; \Delta, z:C \oplus D, x:A \otimes B \vdash z \triangleright \{x(y).P, x(y).Q\} \simeq_c x(y).z \triangleright \{P, Q\} :: k:E$	(II-13)
$\Gamma; \Delta, x:A \& B, y:C \& D \vdash x \triangleleft 1'_i; y \triangleleft 1_i; P \simeq_c y \triangleleft 1_i; x \triangleleft 1'_i; P :: k:E$	(II-14)
$\Gamma; \Delta, x:A \oplus B, y:C \& D \vdash x \triangleright \{y \triangleleft 1_i; P, y \triangleleft 1_i; Q\} \simeq_c y \triangleleft 1_i; x \triangleright \{P, Q\} :: k:E$	(II-15)
$\Gamma, u:C; \Delta, z:A \& B \vdash z \triangleleft 1_i; \overline{u}(y).P \simeq_c \overline{u}(y).z \triangleleft 1_i; P :: k:E$	(II-16)
$\Gamma; \Delta, z:C \& D, x:A \multimap B \vdash z \triangleleft 1_i; \overline{x}(y).(P \mid Q) \simeq_c \overline{x}(y).(z \triangleleft 1_i; P \mid Q) :: k:E$	(II-17)
$\Gamma; \Delta, z:C \& D, x:A \multimap B \vdash z \triangleleft 1_i; \overline{x}(y).(P \mid Q) \simeq_c \overline{x}(y).(P \mid z \triangleleft 1_i; Q) :: k:E$	(II-18)
$\Gamma; \Delta, z:C \& D, x:A \otimes B \vdash z \triangleleft 1_i; x(y).P \simeq_c x(y).z \triangleleft 1_i; P :: k:E$	(II-19)

Figure 7. Process equalities induced by proof conversions, second kind (Part II - cf. Def. A.4).

(id) $(\nu x)([x \leftrightarrow y] \mid P) \xrightarrow{\tau} P\{y/x\}$	(rep) $!x(y).P \xrightarrow{x(z)} P\{z/y\} \mid !x(y).P$			
(n.out) $\overline{x}y.P \xrightarrow{\overline{x}y} P$	(n.in) $x(y).P \xrightarrow{x(z)} P\{z/y\}$	(s.out) $x \triangleleft 1; P \xrightarrow{x \triangleleft 1} P$	(s.in) $x \triangleright \{1_i : P_i\}_{i \in I} \xrightarrow{x \triangleleft 1_j} P_j \quad (j \in I)$	
(par) $\frac{P \xrightarrow{\lambda} Q}{P \mid R \xrightarrow{\lambda} Q \mid R}$	(com) $\frac{P \xrightarrow{\overline{\lambda}} P' \quad Q \xrightarrow{\lambda} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'}$	(res) $\frac{P \xrightarrow{\lambda} Q}{(\nu y)P \xrightarrow{\lambda} (\nu y)Q}$	(open) $\frac{P \xrightarrow{\overline{x}y} Q}{(\nu y)P \xrightarrow{\overline{x}(y)} Q}$	(close) $\frac{P \xrightarrow{\overline{x}(y)} P' \quad Q \xrightarrow{x(y)} Q'}{P \mid Q \xrightarrow{\tau} (\nu y)(P' \mid Q')}$

Figure 8. LTS for Processes.

Actions are name input $x(y)$, the offer $x \triangleleft 1$, and their matching co-actions, respectively the output $\overline{x}y$ and bound output $\overline{x}(y)$ actions, and the selection $x \triangleleft 1$. The bound output $\overline{x}(y)$ denotes extrusion of a fresh name y along x . Internal action is denoted by τ . In general, an action requires a matching co-action in the environment to enable progress.

Definition A.5 (Labeled Transition System). *The relation labeled transition ($P \xrightarrow{\lambda} Q$) is defined by the rules in Fig. 8, subject to the side conditions: in rule (res), we require $y \notin \text{fn}(\lambda)$; in rule (par), we require $\text{bn}(\lambda) \cap \text{fn}(R) = \emptyset$; in rule (close), we require $y \notin \text{fn}(Q)$. We omit the symmetric versions of rules (par), (com), and (close).*

We write $\text{subj}(\lambda)$ for the subject of the action λ , that is, the channel along which the action takes place. Weak transitions are defined as usual. Let us write $\rho_1 \rho_2$ for the composition of relations ρ_1, ρ_2 and \Rightarrow for the reflexive, transitive closure of $\xrightarrow{\tau}$. Notation $\xRightarrow{\lambda}$ stands for $\Rightarrow \xrightarrow{\lambda} \Rightarrow$ (given $\lambda \neq \tau$) and $\xRightarrow{\tau}$ stands for \Rightarrow . We recall basic facts about reduction, structural congruence, and labeled transition: closure of labeled transitions under structural congruence, and coincidence of τ -labeled transition and reduction [27]: (1) if $P \equiv \xrightarrow{\lambda} Q$ then $P \xrightarrow{\lambda} \equiv Q$; (2) $P \rightarrow Q$ iff $P \xrightarrow{\tau} \equiv Q$.

LTS for Global Types With Recursion. We define an LTS over extended global types, which adapts the LTS in [12], with refined intermediate steps. We first define the set of observables:

$$\sigma ::= p \mid p \triangleleft 1 \mid \overline{p} \mid \overline{p \triangleleft 1}$$

Definition A.6 (LTS on Global Types). *The relation labeled transition over global types ($G \xrightarrow{\sigma} G'$) is defined by the rules in Figure 9.*

Relating Labels. We now relate the labels for a global type G and the labels for processes in $\mathcal{S}^k(G)$. The latter are defined as follows, with associated transition rules as in, e.g., [3]:

$$\lambda ::= \tau \mid x(y) \mid x \triangleleft 1 \mid \overline{x}y \mid \overline{x}(y) \mid \overline{x \triangleleft 1}$$

This relation is tight, with minor differences on output actions:

$$\begin{array}{l}
\text{(G1)} \quad p \twoheadrightarrow q: \{1_i \langle U_i \rangle . G_i\}_{i \in I} \xrightarrow{\overline{p \triangleleft 1_j}} p \rightsquigarrow q: 1_j \langle U_j \rangle . G_j \quad (j \in I) \\
\text{(G2)} \quad p \rightsquigarrow q: 1 \langle U \rangle . G \xrightarrow{\overline{p}} p \rightsquigarrow q: 1 \langle (U) \rangle . G \quad \text{(G3)} \quad p \rightsquigarrow q: 1 \langle (U) \rangle . G \xrightarrow{q \triangleleft 1} p \rightsquigarrow q: \langle (U) \rangle . G \\
\text{(G4)} \quad \frac{}{p \rightsquigarrow q: \langle (U) \rangle . G \xrightarrow{q} G} \quad \text{(G5)} \quad \frac{G \{ \mu \mathcal{X} . G / \mathcal{X} \} \xrightarrow{\sigma} G'}{\mu \mathcal{X} . G \xrightarrow{\sigma} G'}
\end{array}$$

Figure 9. LTS over (Extended) Global Types

Definition A.7. Let k and p be a name and a participant identity, respectively. The mapping $\llbracket \cdot \rrbracket^k$ from global type labels σ to process labels λ , and the mapping $\llbracket \cdot \rrbracket_p$ from process labels λ to global type labels σ are defined inductively as follows:

$$\begin{array}{ll}
\llbracket p \rrbracket^k = k & \llbracket k \rrbracket_p = p \\
\llbracket p \triangleleft 1 \rrbracket^k = k \triangleleft 1 & \llbracket k \triangleleft 1 \rrbracket_p = p \triangleleft 1 \\
\llbracket \overline{p} \rrbracket^k = \overline{k} & \llbracket \overline{k} \rrbracket_p = \overline{p} \\
\llbracket p \triangleleft 1 \rrbracket^k = \overline{k} \triangleleft 1 & \llbracket k \triangleleft 1 \rrbracket_p = p \triangleleft 1
\end{array}$$

Extended Global Types and Annotated Mediums. To establish operational correspondence, we consider *extended* global types, defined as follows:

$$\begin{array}{l}
G ::= \text{end} \mid p \twoheadrightarrow q: \{1_i \langle U_i \rangle . G_i\}_{i \in I} \mid \mu \mathcal{X} . G \mid \mathcal{X} \\
\mid p \rightsquigarrow q: 1 \langle U \rangle . G \mid p \rightsquigarrow q: 1 \langle (U) \rangle . G \mid p \rightsquigarrow q: \langle (U) \rangle . G
\end{array}$$

We have introduced three auxiliary forms for global types; denoted with \rightsquigarrow , they represent intermediate steps, as we describe next.

First, global type $p \rightsquigarrow q: 1 \langle U \rangle . G$ denotes the commitment of p to *output* along label 1. The global type $p \rightsquigarrow q: 1 \langle (U) \rangle . G$ denotes the commitment of q to *input* along 1. Finally, type $p \rightsquigarrow q: \langle (U) \rangle . G$ represents the state just before the actual input action by q . The definition of annotated mediums (Def. 4.4) is extended as well:

- $\mathcal{M}^\mu \llbracket p \rightsquigarrow q: 1 \langle U \rangle . G \rrbracket_k = c_p(u). \widehat{k}. (c_q \triangleleft 1; k \triangleright \{1 : \overline{c_q}(v). ([u \leftrightarrow v] \mid k. \mathcal{M}^\mu \llbracket G \rrbracket_k)\})$
- $\mathcal{M}^\mu \llbracket p \rightsquigarrow q: 1 \langle (U) \rangle . G \rrbracket_k = c_q \triangleleft 1; k \triangleright \{1 : \overline{c_q}(v). ([u \leftrightarrow v] \mid k. \mathcal{M}^\mu \llbracket G \rrbracket_k)\}$
- $\mathcal{M}^\mu \llbracket p \rightsquigarrow q: \langle (U) \rangle . G \rrbracket_k = \overline{c_q}(v). ([u \leftrightarrow v] \mid k. \mathcal{M}^\mu \llbracket G \rrbracket_k)$

B. Omitted Proofs from § 4.2

B.1 Relating SWF Global Types and Typed Mediums

Following Remark 4.11, here we consider the analogous of Theorems 4.8 and 4.10 but in the setting of global types which are SWF.

Proposition B.1. Let $G = G_1 \mid G_2$ be a global type in \mathcal{G}^{fn} . If $\Gamma; \Delta \vdash M[G]$ is a compositional typing then there exist disjoint Δ_1, Δ_2 such that: (i) $\Delta = \Delta_1 \cup \Delta_2$, and (ii) $\Gamma; \Delta_1 \vdash M[G_1]$, and $\Gamma; \Delta_2 \vdash M[G_2]$ are compositional typings.

Proof. By inversion on typing. By assumption the typing for $M[G]$ is a compositional one; hence, there is a $c_j: A_j \in \Delta$ for each $r_j \in \text{part}(G) = \text{part}(G_1) \cup \text{part}(G_2)$. This means that, necessarily, the typed parallel composition between $M[G_1]$ and $M[G_2]$ is independent (in the sense of the derived rule (INDCOMP) in § A.1). In fact, a non independent composition (i.e., using rule (Tcut)) would contradict the assumption that $c_j: A_j \in \Delta$ for each $r_j \in \text{part}(G) = \text{part}(G_1) \cup \text{part}(G_2)$, for there would have to exist a participant $r_k \in \text{part}(G_1) \cup \text{part}(G_2)$ but without a $c_k: A_k \in \Delta$. \square

B.1.1 SWF Global Types Ensure Compositional Typings

We state and prove Theorem B.3, the analogous of Theorem 4.8. The proof uses the following auxiliary proposition, whose proof follows by construction.

Proposition B.2. If $p_1 \twoheadrightarrow p_2: \{1_i \langle U_i \rangle . G_i\}_{i \in I}$ is SWF in \mathcal{G}^{fn} then all G^i ($i \in I$) are SWF too. Also, if $G = G_1 \mid G_2$ is SWF then G_j ($j \in \{1, 2\}$) are SWF too.

Theorem B.3. Let $G \in \mathcal{G}^{fn}$ be a global type, with $\text{part}(G) = \{p_1, \dots, p_n\}$. If G is SWF then

$$\Gamma; c_1: \langle G \wr p_1 \rangle, \dots, c_n: \langle G \wr p_n \rangle \vdash M[G]$$

is a compositional typing, for some Γ .

Proof. By induction on the structure of G , relying on simple projection (Def. A.1).

- (Case $G = \text{end}$): Then the thesis holds vacuously, for $\text{part}(G) = \emptyset$.

$$\begin{array}{c}
\frac{\Gamma; u : \langle\langle U_1 \rangle\rangle \vdash [u \leftrightarrow v] :: v : \langle\langle U_1 \rangle\rangle \quad (\text{Tid}) \quad \frac{\Gamma; c_p : \langle\langle G^1 \wr p \rangle\rangle, c_q : \langle\langle G^1 \wr q \rangle\rangle, \Delta_1 \vdash M[G^1] \quad (\text{T1L})}{\Gamma; c_p : \langle\langle G^1 \wr p \rangle\rangle, c_q : \langle\langle G^1 \wr q \rangle\rangle, \Delta_1 \vdash \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \quad (\text{T}\rightarrow\text{L})}}{\Gamma; u : \langle\langle U_1 \rangle\rangle, c_p : \langle\langle G^1 \wr p \rangle\rangle, c_q : \langle\langle U_1 \rangle\rangle \multimap \langle\langle G^1 \wr q \rangle\rangle, \Delta_1 \vdash \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \quad (\text{T}\&\text{L}_1)} \\
\frac{\Gamma; u : \langle\langle U_1 \rangle\rangle, c_p : \langle\langle G^1 \wr p \rangle\rangle, c_q : \&\{1_1 : \langle\langle U_1 \rangle\rangle \multimap \langle\langle G^1 \wr q \rangle\rangle\}_{1\}, \Delta_1 \vdash c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \quad (\text{T}\&\text{L}_2)}{\Gamma; u : \langle\langle U_1 \rangle\rangle, c_p : \langle\langle G^1 \wr p \rangle\rangle, c_q : \&\{1_i : (\langle\langle U_i \rangle\rangle \multimap \langle\langle G^i \wr q \rangle\rangle)\}_{i \in I}, \Delta_1 \vdash c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \quad (\text{T}\otimes\text{L}_2)} \\
\frac{\Gamma; c_p : \langle\langle U_1 \rangle\rangle \otimes \langle\langle G^1 \wr p \rangle\rangle, c_q : \&\{1_i : (\langle\langle U_i \rangle\rangle \multimap \langle\langle G^i \wr q \rangle\rangle)\}_{i \in I}, \Delta_1 \vdash c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \quad (\text{T}\otimes\text{L})}
\end{array}$$

Figure 10. Derivation for $c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1])$ (cf. (17) in Page 17)

- (Case $G = p_1 \rightarrow p_2 : \{1_i \langle U_i \rangle . G^i\}_{i \in I}$): By the well-formedness assumption (Def. A.2), local types $G \wr p_1, \dots, G \wr p_n$ are all defined. Writing p and q instead of p_1 and p_2 , by Def. A.1 we have:

$$G \wr p = p! \{1_i \langle U_i \rangle . G^i \wr p\}_{i \in I} \quad (7)$$

$$G \wr q = p? \{1_i \langle U_i \rangle . G^i \wr q\}_{i \in I} \quad (8)$$

$$G \wr p_j = G^j \wr p_j \quad \text{for every } j \in \{3, \dots, n\} \quad (9)$$

In (9), it is useful to recall that Def. A.1 decrees that, for every $j \in \{3, \dots, n\}$,

$$G^l \wr p_j = G^k \wr p_j, \quad \text{for any } l, k \in I \quad (10)$$

which explains why taking $G^1 \wr p_j$ is enough. We need to show that:

$$\Gamma; c_p : \langle\langle G \wr p \rangle\rangle, c_q : \langle\langle G \wr q \rangle\rangle, c_{p_3} : \langle\langle G^1 \wr p_3 \rangle\rangle, \dots, c_{p_n} : \langle\langle G^1 \wr p_n \rangle\rangle \vdash M[G] \quad (11)$$

is a compositional typing, for some Γ . To improve readability, and without loss of generality, we describe the case $I = \{1, 2\}$. By Def. 4.1 we have:

$$M[G] = c_p \triangleright \begin{cases} 1_1 : c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \\ 1_2 : c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2]) \end{cases}$$

and by combining (7) and (8) with Def. 4.7 we have:

$$\langle\langle G \wr p \rangle\rangle = \oplus \{1_1 : \langle\langle U_1 \rangle\rangle \otimes \langle\langle G^1 \wr p \rangle\rangle, 1_2 : U_2 \otimes \langle\langle G^2 \wr p \rangle\rangle\}_{1,2} \quad (12)$$

$$\langle\langle G \wr q \rangle\rangle = \&\{1_1 : \langle\langle U_1 \rangle\rangle \multimap \langle\langle G^1 \wr q \rangle\rangle, 1_2 : U_2 \multimap \langle\langle G^2 \wr q \rangle\rangle\}_{1,2} \quad (13)$$

Now, by assumption G is SWF; by Prop. B.2, then also both its sub-types G^1 and G^2 are SWF. Thus, using IH twice we may infer that both

$$\Gamma; c_p : \langle\langle G^1 \wr p \rangle\rangle, c_q : \langle\langle G^1 \wr q \rangle\rangle, \underbrace{c_{p_3} : \langle\langle G^1 \wr p_3 \rangle\rangle, \dots, c_{p_n} : \langle\langle G^1 \wr p_n \rangle\rangle}_{\Delta_1} \vdash M[G^1] \quad (14)$$

$$\Gamma; c_p : \langle\langle G^2 \wr p \rangle\rangle, c_q : \langle\langle G^2 \wr q \rangle\rangle, \underbrace{c_{p_3} : \langle\langle G^2 \wr p_3 \rangle\rangle, \dots, c_{p_n} : \langle\langle G^2 \wr p_n \rangle\rangle}_{\Delta_2} \vdash M[G^2] \quad (15)$$

are compositional typings for some Γ . Now, using (10) we infer $G^1 \wr p_j = G^2 \wr p_j$, for all $j \in \{3, \dots, n\}$. Therefore,

$$\Delta_1 = \Delta_2. \quad (16)$$

Using (14), the derivation for

$$\Gamma; c_p : \langle\langle U_1 \rangle\rangle \otimes \langle\langle G^1 \wr p \rangle\rangle, c_q : \&\{1_1 : \langle\langle U_1 \rangle\rangle \multimap \langle\langle G^1 \wr q \rangle\rangle, 1_2 : \langle\langle U_2 \rangle\rangle \multimap \langle\langle G^2 \wr q \rangle\rangle\}_{1,2}, \Delta_1 \vdash c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \quad (17)$$

is as in Fig. 10. Using (15), the derivation for

$$\Gamma; c_p : \langle\langle U_2 \rangle\rangle \otimes \langle\langle G^2 \wr p \rangle\rangle, c_q : \&\{1_1 : \langle\langle U_1 \rangle\rangle \multimap \langle\langle G^1 \wr q \rangle\rangle, 1_2 : \langle\langle U_2 \rangle\rangle \multimap \langle\langle G^2 \wr q \rangle\rangle\}_{1,2}, \Delta_2 \vdash c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2]) \quad (18)$$

is obtained in an analogous way. We now have all requirements for completing the desired typing. Using rule (T \oplus L) (cf. Fig. 3) with (17) and (18) as premises, and crucially relying on (16), we may derive:

$$\Gamma; c_p : \oplus \{1_1 : \langle\langle U_1 \rangle\rangle \otimes \langle\langle G^1 \wr p \rangle\rangle, 1_2 : \langle\langle U_2 \rangle\rangle \otimes \langle\langle G^2 \wr p \rangle\rangle\}_{1,2}, c_q : \&\{1_1 : \langle\langle U_1 \rangle\rangle \multimap \langle\langle G^1 \wr q \rangle\rangle, 1_2 : \langle\langle U_2 \rangle\rangle \multimap \langle\langle G^2 \wr q \rangle\rangle\}_{1,2}, \Delta_1 \vdash M[G]$$

which is easily seen to be a compositional typing.

- (Case $G = G_1 \mid G_2$): By Def. 3.1, we know that $\text{part}(G) = \text{part}(G_1) \cup \text{part}(G_2)$. Let us write $\text{part}(G_1) = \{p_1, \dots, p_k\}$ and $\text{part}(G_2) = \{p_{k+1}, \dots, p_n\}$. By assumption G is SWF; by Prop. B.2, then also G_1 and G_2 are SWF. Thus, using IH twice we may

infer that both

$$\Gamma; c_1:\langle\langle G_1 \wr p_1 \rangle\rangle, \dots, c_k:\langle\langle G_1 \wr p_k \rangle\rangle \vdash M[G_1] \quad (19)$$

$$\Gamma; c_{k+1}:\langle\langle G_2 \wr p_{k+1} \rangle\rangle, \dots, c_n:\langle\langle G_2 \wr p_n \rangle\rangle \vdash M[G_2] \quad (20)$$

are compositional typings for some Γ . Now, by Def. A.1, we infer that for all $p_i \in \text{part}(G)$ then either $G_1 \wr p_i$ is defined or $G_2 \wr p_i$ is defined, but not both. We therefore infer that $\Delta_1 \# \Delta_2$. Then, using independent parallel composition (cf. rule (INDCOMP) in § A.1) we may infer the typing:

$$\Gamma; \Delta_1, \Delta_2 \vdash M[G_1] \mid M[G_2]$$

Hence, since by Def. 4.1 $M[G] = M[G_1] \mid M[G_2]$, the thesis follows. \square

B.1.2 Compositional Typings Induce SWF Global Types

We state and prove the analogous of Theorem 4.10 for SWF global types.

Theorem B.4. *Let $G \in \mathcal{G}^{\text{fin}}$ be a global type. If*

$$\Gamma; c_1:A_1, \dots, c_n:A_n \vdash M[G]$$

is a compositional typing for $M[G]$ then $\exists T_1, \dots, T_n$ s.t. $G \wr p_j \preceq^{\sqcup} T_j$ and $\langle\langle T_j \rangle\rangle = A_j$, for all $p_j \in G$.

Proof. By induction on the structure of G :

- (Case $G = \text{end}$): Then $M[G] = \mathbf{0}$, $\text{part}(G) = \emptyset$, and the thesis follows vacuously. Notice that from the assumption $\Gamma; \cdot \vdash M[G]$ and rule (T1L) we may derive $\Gamma; c_j:\mathbf{1} \vdash M[G]$, for any name c_j . In such a case, we observe that Def. A.1 decrees that $\text{end} \wr p_j = \text{end}$, for any p_j . The thesis holds, for $\langle\langle \text{end} \rangle\rangle = \mathbf{1}$.
- (Case $G = p_1 \rightarrow p_2: \{1_i \langle U_i \rangle . G^i\}_{i \in I}$): Then $\text{part}(G) = \{p_1, \dots, p_n\}$, with $n \geq 2$. By Def. A.1, we have that

$$G \wr p_1 = p_1! \{1_i \langle U_i \rangle . G^i \wr p_1\}_{i \in I} \quad (21)$$

$$G \wr p_2 = p_1? \{1_i \langle U_i \rangle . G^i \wr p_2\}_{i \in I} \quad (22)$$

$$G \wr p_j = G^1 \wr p_j, \quad \text{for all } p_j \in \{p_3, \dots, p_n\} \quad (23)$$

Without loss of generality, we describe the case $I = \{1, 2\}$. Writing p and q instead of p_1 and p_2 , by expanding Def. 4.1 we obtain:

$$M[G] = c_p \triangleright \begin{cases} 1_1 : c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \\ 1_2 : c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2]) \end{cases} \quad (24)$$

while by assumption we have the compositional typing

$$\Gamma; c_p:A_1, c_q:A_2, c_{p3}:A_3, \dots, c_{pn}:A_n \vdash M[G] \quad (25)$$

We must exhibit local types T_1, \dots, T_n such that, for all $i \in \{1, \dots, n\}$:

(i) $G \wr p_i \preceq^{\sqcup} T_i$ and $A_i = \langle\langle T_i \rangle\rangle$.

First, by inversion on typing on (24) and (25), we infer that there exist binary session types $C_1, C_2, D_1, D_2, \dots, D_k$, and U_1, U_2, \dots, U_k such that

$$A_1 = \oplus \{1_1 : U_1 \otimes C_1, 1_2 : U_2 \otimes C_2\} \quad (26)$$

$$A_2 = \& \{1_1 : U_1 \multimap D_1, 1_2 : U_2 \multimap D_2, 1_3 : U_3 \multimap D_3, \dots, 1_k : U_k \multimap D_k\} \quad (27)$$

In (27), $1_3 : U_3 \multimap D_3, \dots, 1_k : U_k \multimap D_k$ correspond to labelled alternatives that may be silently added by rule (T&L₂). Now, using rule (T⊕L):

$$\Gamma; c_p:U_1 \otimes C_1, c_q:A_2, \Delta \vdash c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \quad (*)$$

$$\Gamma; c_p:U_2 \otimes C_2, c_q:A_2, \Delta \vdash c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2]) \quad (**)$$

$$\Gamma; c_p:A_1, c_q:A_2, \Delta \vdash M[G] :: - : \mathbf{1}$$

where $\Delta = c_{p3}:A_3, \dots, c_{pn}:A_n$, i.e., Δ collects typings for participants not involved in the exchange. Notice that the assumption of compositional typing (in particular, the fact that $M[G]$ does not offer any behavior on the right-hand side typing) is crucial in the above inversion. By further inversion on typing, we may infer typings for $M[G^1]$ and $M[G^2]$:

$$\Gamma; c_p:C_1, c_q:D_1, \Delta \vdash M[G^1] \quad (28)$$

$$\Gamma; c_p:C_2, c_q:D_2, \Delta \vdash M[G^2] \quad (29)$$

This way, e.g., the derivation for (29) is below, based on premise (**) above: where we have denoted explicitly the several possible uses of silent rule (T&L₂). It is easy to see that (28) and (29) are compositional typings. Then, IH ensures the existence of local types $R_1, \dots, R_n, S_1, \dots, S_n$ such that:

$$G^1 \wr p_j \preceq^{\sqcup} R_j \text{ and } G^2 \wr p_j \preceq^{\sqcup} S_j, \quad \text{for all } j \in \{1, \dots, n\}$$

In particular, $\langle\langle R_1 \rangle\rangle = C_1$, $\langle\langle R_2 \rangle\rangle = D_1$, $\langle\langle S_1 \rangle\rangle = C_2$, and $\langle\langle S_2 \rangle\rangle = D_2$.

We notice that Δ is always kept unchanged in the typing derivations for (28) and (29). Therefore, for all $k \in \{3, \dots, n\}$, we have:

$$A_k = \langle\langle R_k \rangle\rangle = \langle\langle S_k \rangle\rangle \quad \text{and} \quad R_k = S_k \quad (30)$$

In turn, by combining (23) and (30) we infer the thesis for participants p_3, \dots, p_n :

$$G \upharpoonright p_k \preceq^\sqcup R_k = T_k, \quad \text{for all } k \in \{3, \dots, n\}$$

We are thus left to show the thesis for p_1 and p_2 . We first establish T_1 and T_2 by building upon local types R_1, R_2, S_1, S_2 (just established), following the typing derivation for

$$c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2])$$

(shown above) and for

$$c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1])$$

(which is built analogously). We thus have:

$$\begin{aligned} T_1 &= p_1! \{1_1 \langle U_1 \rangle . R_1, 1_2 \langle U_2 \rangle . S_1\} \\ T_2 &= p_1? \{1_1 \langle U_1 \rangle . R_2, 1_2 \langle U_2 \rangle . S_2, 1_3 \langle U_3 \rangle . S_3, \dots, 1_k \langle U_k \rangle . S_k\} \end{aligned}$$

and using (21), (22), and Definitions 4.9 and 4.7, we may verify that:

(i) $G \upharpoonright p_1 = T_1$, $G \upharpoonright p_2 \preceq^\sqcup T_2$ and (ii) $\langle\langle T_1 \rangle\rangle = A_1$ and $\langle\langle T_2 \rangle\rangle = A_2$.

- (Case $G = G_1 \mid G_2$): Then $\text{part}(G) = \{p_1, \dots, p_n\}$, with $n \geq 2$. Recall that $\text{part}(G_1 \mid G_2) = \text{part}(G_1) \cup \text{part}(G_2)$. By Def. 4.1 we may state the compositional typing assumption as

$$\Gamma; \Delta \vdash M[G_1] \mid M[G_2]$$

where $\Delta = c_1:A_1, \dots, c_n:A_n$, with a c_j for each $r_j \in \text{part}(G_1 \mid G_2)$ (with $j \in \{1, \dots, n\}$). Let $\text{part}(G_1) = \{p_1, \dots, p_k\}$ and $\text{part}(G_2) = \{p_{k+1}, \dots, p_n\}$. By Prop. B.1, there exist disjoint Δ_1, Δ_2 such that $\Delta = \Delta_1 \cup \Delta_2$ and the compositional typings

$$\begin{aligned} &\Gamma; \underbrace{c_1:A_1, \dots, c_k:A_k}_{\Delta_1} \vdash M[G_1] \\ &\Gamma; \underbrace{c_{k+1}:A_{k+1}, \dots, c_n:A_n}_{\Delta_2} \vdash M[G_2] \end{aligned}$$

hold. We may then apply IH on both $M[G_1]$ and $M[G_2]$, and so infer local types R_1, \dots, R_k and S_{k+1}, \dots, S_n such that (i) $G_1 \upharpoonright p_i \preceq^\sqcup R_i$ and (ii) $A_i = \langle\langle R_i \rangle\rangle$ (with $i \in \{1, \dots, k\}$) and (iii) $G_1 \upharpoonright p_h \preceq^\sqcup S_h$ (iv) $A_h = \langle\langle S_h \rangle\rangle$ (with $h \in \{k+1, \dots, n\}$). This is enough to conclude the thesis, for Def. A.1 says that parallel global types do not share participants. Therefore, for every $p_l \in \{p_1, \dots, p_n\}$ then either $G \upharpoonright p_l = G_1 \upharpoonright p_l$ or $G \upharpoonright p_l = G_2 \upharpoonright p_l$.

□

B.2 Proof of Theorem 4.8: MWF Global Types Ensure Compositional Typings

We first state the analogous of Prop. B.2:

Proposition B.5. *If $p_1 \rightarrow p_2: \{1_i \langle U_i \rangle . G^i\}_{i \in I}$ is MWF in \mathcal{G}^{fin} then all G^i ($i \in I$) are MWF too. Also, if $G = G_1 \mid G_2$ is MWF then G_j ($j \in \{1, 2\}$) are MWF too.*

We now repeat the statement in Page 7:

Theorem B.6 (4.8). *Let $G \in \mathcal{G}^{\text{fin}}$ be a global type, with $\text{part}(G) = \{p_1, \dots, p_n\}$.*

If G is MWF then $\Gamma; c_1: \langle\langle G \upharpoonright p_1 \rangle\rangle, \dots, c_n: \langle\langle G \upharpoonright p_n \rangle\rangle \vdash M[G]$ is a compositional typing, for some Γ .

Proof. By induction on the structure of G . The most interesting case is when $G = p_1 \rightarrow p_2: \{1_i \langle U_i \rangle . G^i\}_{i \in I}$. Remaining cases are as in the proof of Theorem B.3 (Page 16).

By the well-formedness assumption (Def. 3.4), local types $G \upharpoonright p_1, \dots, G \upharpoonright p_n$ are all defined. Writing p and q instead of p_1 and p_2 , by Def. 3.3 we have:

$$G \upharpoonright p = p! \{1_i \langle U_i \rangle . G^i \upharpoonright p\}_{i \in I} \quad (31)$$

$$G \upharpoonright q = p? \{1_i \langle U_i \rangle . G^i \upharpoonright q\}_{i \in I} \quad (32)$$

$$G \upharpoonright p_j = \sqcup_{i \in I} G^i \upharpoonright p_j \quad \text{for every } j \in \{3, \dots, n\} \quad (33)$$

We need to show that, for some Γ ,

$$\Gamma; c_p: \langle\langle G \upharpoonright p \rangle\rangle, c_q: \langle\langle G \upharpoonright q \rangle\rangle, c_{p_3}: \langle\langle G \upharpoonright p_3 \rangle\rangle, \dots, c_{p_n}: \langle\langle G \upharpoonright p_n \rangle\rangle \vdash M[G] \quad (34)$$

is a compositional typing. Without loss of generality, we detail the case $I = \{1, 2\}$. By Def. 4.1, we have:

$$M[G] = c_p \triangleright \begin{cases} 1_1 : c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \\ 1_2 : c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2]) \end{cases}$$

and by combining (31) and (32) with Def. 4.7 we have:

$$\begin{aligned}\langle\!\langle G \upharpoonright \mathbf{p} \rangle\!\rangle &= \oplus\{1_1 : \langle\!\langle U_1 \rangle\!\rangle \otimes \langle\!\langle G^1 \upharpoonright \mathbf{p} \rangle\!\rangle, 1_2 : \langle\!\langle U_2 \rangle\!\rangle \otimes \langle\!\langle G^2 \upharpoonright \mathbf{p} \rangle\!\rangle\}_{i \in I} \\ \langle\!\langle G \upharpoonright \mathbf{q} \rangle\!\rangle &= \&S\{1_1 : \langle\!\langle U_1 \rangle\!\rangle \multimap \langle\!\langle G^1 \upharpoonright \mathbf{q} \rangle\!\rangle, 1_2 : \langle\!\langle U_2 \rangle\!\rangle \multimap \langle\!\langle G^2 \upharpoonright \mathbf{q} \rangle\!\rangle\}_{i \in I}\end{aligned}$$

Now, by assumption G is MWF; then, by Prop. B.5, both G^1 and G^2 are MWF too. Therefore, by using IH twice we may infer that both

$$\Gamma; c_p : \langle\!\langle G^1 \upharpoonright \mathbf{p} \rangle\!\rangle, c_q : \langle\!\langle G^1 \upharpoonright \mathbf{q} \rangle\!\rangle, \underbrace{c_{p_3} : \langle\!\langle G^1 \upharpoonright \mathbf{p}_3 \rangle\!\rangle, \dots, c_{p_n} : \langle\!\langle G^1 \upharpoonright \mathbf{p}_n \rangle\!\rangle}_{\Delta_1} \vdash M[G^1] \quad (35)$$

$$\Gamma; c_p : \langle\!\langle G^2 \upharpoonright \mathbf{p} \rangle\!\rangle, c_q : \langle\!\langle G^2 \upharpoonright \mathbf{q} \rangle\!\rangle, \underbrace{c_{p_3} : \langle\!\langle G^2 \upharpoonright \mathbf{p}_3 \rangle\!\rangle, \dots, c_{p_n} : \langle\!\langle G^2 \upharpoonright \mathbf{p}_n \rangle\!\rangle}_{\Delta_2} \vdash M[G^2] \quad (36)$$

are compositional typings, for any Γ .

Now, to obtain a compositional typing for $M[G]$, we must first address the fact that, differently from what occurs in the proof of Theorem B.3 (cf. equality (16) in Page 17), in this case it is not necessarily the case that Δ_1 and Δ_2 are equal. This discrepancy is due to the merge-based well-formedness assumption, which admits non identical behaviors in branches G^1 and G^2 in the case of (local) branching types.

We proceed by induction on k , defined as the size of Δ_1 and Δ_2 (note that $k = n - 2$).

- (Case $k = 0$): Then $\Delta_1 = \Delta_2 = \emptyset$ and \mathbf{p} and \mathbf{q} are the only participants in G . Let us write A_q to stand for the session type

$$\&S\{1_1 : \langle\!\langle U_1 \rangle\!\rangle \multimap \langle\!\langle G^1 \upharpoonright \mathbf{q} \rangle\!\rangle, 1_2 : \langle\!\langle U_2 \rangle\!\rangle \multimap \langle\!\langle G^2 \upharpoonright \mathbf{q} \rangle\!\rangle\}$$

Based on (35) and (36), following the derivation in Fig. 10, we may derive typings

$$\Gamma; c_p : \langle\!\langle U_1 \rangle\!\rangle \otimes \langle\!\langle G^1 \upharpoonright \mathbf{p} \rangle\!\rangle, c_q : A_q \vdash c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \quad (37)$$

$$\Gamma; c_p : \langle\!\langle U_2 \rangle\!\rangle \otimes \langle\!\langle G^2 \upharpoonright \mathbf{p} \rangle\!\rangle, c_q : A_q \vdash c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2]) \quad (38)$$

Then, using (37) and (38) we may derive, using rule (T \oplus L):

$$\Gamma; c_p : \&S\{1_1 : \langle\!\langle U_1 \rangle\!\rangle \otimes \langle\!\langle G^1 \upharpoonright \mathbf{p} \rangle\!\rangle, 1_2 : \langle\!\langle U_2 \rangle\!\rangle \otimes \langle\!\langle G^2 \upharpoonright \mathbf{p} \rangle\!\rangle\}, c_q : A_q \vdash M[G]$$

and the thesis follows.

- (Case $k > 0$): Then there exists a participant \mathbf{p}_k , types $B_1 = \langle\!\langle G^1 \upharpoonright \mathbf{p}_k \rangle\!\rangle$, $B_2 = \langle\!\langle G^2 \upharpoonright \mathbf{p}_k \rangle\!\rangle$ and environments Δ'_1, Δ'_2 such that $\Delta_1 = c_{p_k} : B_1, \Delta'_1$ and $\Delta_2 = c_{p_k} : B_2, \Delta'_2$.

By induction hypothesis, there is a compositional typing starting from

$$\begin{aligned}\Gamma; c_p : \langle\!\langle G^1 \upharpoonright \mathbf{p} \rangle\!\rangle, c_q : \langle\!\langle G^1 \upharpoonright \mathbf{q} \rangle\!\rangle, \Delta'_1 &\vdash M[G^1] \\ \Gamma; c_p : \langle\!\langle G^2 \upharpoonright \mathbf{p} \rangle\!\rangle, c_q : \langle\!\langle G^2 \upharpoonright \mathbf{q} \rangle\!\rangle, \Delta'_2 &\vdash M[G^2]\end{aligned}$$

resulting into

$$\Gamma; c_p : \&S\{1_1 : \langle\!\langle U_1 \rangle\!\rangle \otimes \langle\!\langle G^1 \upharpoonright \mathbf{p} \rangle\!\rangle, 1_2 : \langle\!\langle U_2 \rangle\!\rangle \otimes \langle\!\langle G^2 \upharpoonright \mathbf{p} \rangle\!\rangle\}, c_q : A_q, \Delta'_1 \vdash M[G]$$

since $\Delta'_1 = \Delta'_2$. To extend the typing derivation to Δ_1 and Δ_2 , we proceed by a case analysis on the shape of B_1 and B_2 . We aim to show that (a) B_1 and B_2 are already identical session types or (b) that typing allows us to transform them into identical types. We rely on the definition of \sqcup (Def. 3.2). There are three cases:

- (1) Case $B_1 = 1$: Then, since Def. 3.2 decrees $\text{end} \sqcup \text{end} = \text{end}$ and the fact that merge-based well-definedness depends on \sqcup , we may infer $B_2 = 1$. Hence, $\Delta_1 = \Delta_2$ and the desired derivation is obtained as in the base case.
- (2) Case $B_1 = \oplus\{1_k : A_k\}_{k \in K}$: Then, similarly as in the previous sub case, by Def. 3.2 we immediately infer that $B_2 = \oplus\{1_k : A_k\}_{k \in K}$. We this may infer that $\Delta_1 = \Delta_2$ and complete the derivation.
- (3) Case $B_1 = \&S\{1_h : A_h\}_{h \in H}$: This is the interesting case, for even if merge-based well-formedness of G ensures that both B_1 and B_2 are identical.

If B_1 and B_2 are identical then we proceed as in previous sub cases. Otherwise, then due to \sqcup there are some labeled alternatives in B_1 but not in B_2 and/or viceversa. Also, Def. 3.2 ensures that common options (if any) are identical in both branches. We may then use the rule (T $\&$ L $_2$) to “complement” occurrences of types B_1 and B_2 in (35) and (36) with appropriate options, so as to make them coincide and achieve identical typing. This rule is silent; as labels are finite, this completing task is also finite, and results into $\Delta_1 = \Delta_2$. We then may complete the derivation as in the base case. This concludes the proof. \square

B.3 Proof of Theorem 4.10: Compositional Typings Induce MWF Global Types

We repeat the statement given in Page 8:

Theorem B.7 (4.10). *Let $G \in \mathcal{G}^{\text{fm}}$ be a global type. If*

$$\Gamma; c_1 : A_1, \dots, c_n : A_n \vdash M[G]$$

is a compositional typing for $M[G]$ then $\exists T_1, \dots, T_n$ s.t. $G \upharpoonright \mathbf{r}_j \preceq^\sqcup T_j$ and $\langle\!\langle T_j \rangle\!\rangle = A_j$, for all $\mathbf{r}_j \in G$.

Proof. By induction on the structure of G :

- (Case $G = \text{end}$): Then $M[G] = \mathbf{0}$, $\text{part}(G) = \emptyset$, and the thesis follows vacuously. Notice that from the assumption $\Gamma; \cdot \vdash M[G]$ and rule (T1L) we may derive $\Gamma; c_j:1 \vdash M[G]$, for any name c_j . In such a case, we observe that Def. 3.3 decrees that $\text{end} \vdash r_j = \text{end}$, for any r_j . The thesis holds, for $\langle\langle \text{end} \rangle\rangle = 1$.
- (Case $G = p_1 \rightarrow p_2: \{1_i \langle U_i \rangle . G^i\}_{i \in I}$): Then $\text{part}(G) = \{p_1, \dots, p_n\}$, with $n \geq 2$. By Def. 3.3, we have that

$$G \upharpoonright p_1 = p_1! \{1_i \langle U_i \rangle . G^i \upharpoonright p_1\}_{i \in I} \quad (39)$$

$$G \upharpoonright p_2 = p_1? \{1_i \langle U_i \rangle . G^i \upharpoonright p_2\}_{i \in I} \quad (40)$$

$$G \upharpoonright p_j = \sqcup_{i \in I} G^i \upharpoonright p_j \quad (41)$$

Without loss of generality, we detail the case $|I| = 2$. Writing p and q instead of p_1 and p_2 , by expanding Def. 4.1 we obtain:

$$M[G] = c_p \triangleright \begin{cases} 1_1 : c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \\ 1_2 : c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2]) \end{cases} \quad (42)$$

while by assumption we have the compositional typing

$$\Gamma; c_p:A_1, c_q:A_2, c_{p3}:A_3, \dots, c_{pn}:A_n \vdash M[G] \quad (43)$$

We must exhibit local types T_1, \dots, T_n such that, for all $i \in \{1, \dots, n\}$:

(i) $G \upharpoonright p_i \preceq^\sqcup T_i$ and $A_i = \langle\langle T_i \rangle\rangle$.

First, by inversion on typing on (42) and (43), we infer that there exist binary session types $C_1, C_2, D_1, D_2, \dots, D_k$, and U_1, U_2, \dots, U_k such that

$$A_1 = \oplus \{1_1 : U_1 \otimes C_1, 1_2 : U_2 \otimes C_2\} \quad (44)$$

$$A_2 = \& \{1_1 : U_1 \multimap D_1, 1_2 : U_2 \multimap D_2, \dots, 1_k : U_k \multimap D_k\} \quad (45)$$

Notice that in (45), we consider labeled alternatives that may be silently added by rule (T&L₂). Now, using rule (T⊕L):

$$\Gamma; c_p:U_1 \otimes C_1, c_q:A_2, \Delta \vdash c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \quad (*)$$

$$\Gamma; c_p:U_2 \otimes C_2, c_q:A_2, \Delta \vdash c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2]) \quad (**)$$

$$\Gamma; c_p:A_1, c_q:A_2, \Delta \vdash M[G] :: - : 1$$

where $\Delta = c_{p3}:A_3, \dots, c_{pn}:A_n$, i.e., Δ collects typings for participants not involved in the exchange. Notice that the assumption of compositional typing (in particular, the fact that $M[G]$ does not offer any behavior on the right-hand side typing) is crucial in the above inversion. By further inversion on typing, we may infer typings for $M[G^1]$ and $M[G^2]$:

$$\Gamma; c_p:C_1, c_q:D_1, \Delta \vdash M[G^1] \quad (46)$$

$$\Gamma; c_p:C_2, c_q:D_2, \Delta \vdash M[G^2] \quad (47)$$

This way, e.g., the derivation for (47) is below, based on premise (**) above:

$$\frac{\frac{\frac{\Gamma; u:U_2 \vdash [u \leftrightarrow v] :: v : U_2 \quad (\text{Tid})}{\Gamma; c_p : C_2, c_q : D_2, \Delta \vdash M[G^2]} \quad (\text{T1L})}{\Gamma; u : U_2, c_p : C_2, c_q : U_2 \multimap D_2, \Delta \vdash \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2])} \quad (\text{T} \multimap \text{L})}{\frac{\Gamma; u : U_2, c_p : C_2, c_q : \& \{1_2 : U_2 \multimap D_2\}_{\{2\}}, \Delta \vdash c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2])}{\Gamma; u : U_2, c_p : C_2, c_q : A_2, \Delta \vdash c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2])} \quad (\text{T} \& \text{L}_2)} \quad (\text{T} \& \text{L}_1)$$

$$\frac{\Gamma; u : U_2, c_p : C_2, c_q : A_2, \Delta \vdash c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2])}{\Gamma; c_p : U_2 \otimes C_2, c_q : A_2, \Delta \vdash c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2])} \quad (\text{T} \otimes \text{L})$$

where we have explicitly denoted the several possible uses of silent rule (T&L₂). It is easy to see that (46) and (47) are compositional typings. Then, IH ensures the existence of local types $R_1, \dots, R_n, S_1, \dots, S_n$ such that:

$$G^1 \upharpoonright p_j \preceq^\sqcup R_j \text{ and } G^2 \upharpoonright p_j \preceq^\sqcup S_j, \quad \text{for all } j \in \{1, \dots, n\}$$

In particular, $\langle\langle R_1 \rangle\rangle = C_1$, $\langle\langle R_2 \rangle\rangle = D_1$, $\langle\langle S_1 \rangle\rangle = C_2$, and $\langle\langle S_2 \rangle\rangle = D_2$.

We notice that Δ remains unchanged in the derivations for (46) and (47). Hence, intuitively, considering merge-based projectability does not play a role in the proof, for our assumption is the compositional typing for $M[G]$. All mergeable branching types for branches of G appear already merged in Δ ; such merged types are propagated in derivations. Therefore, for all $k \in \{3, \dots, n\}$, we have:

$$A_k = \langle\langle R_k \rangle\rangle = \langle\langle S_k \rangle\rangle \quad \text{and} \quad R_k = S_k \quad (48)$$

In turn, by combining (41) and (48) we infer the thesis for participants p_3, \dots, p_n :

$$G \upharpoonright p_k \preceq^\sqcup R_k = T_k, \quad \text{for all } k \in \{3, \dots, n\}$$

We are thus left to show the thesis for p_1 and p_2 . We first establish T_1 and T_2 by building upon local types R_1, R_2, S_1, S_2 (just established), following the typing derivation for

$$c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2])$$

(shown above) and for

$$c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1])$$

(which is built analogously). We thus have:

$$\begin{aligned} T_1 &= \mathbf{p}_1! \{ \mathbf{1}_1 \langle U_1 \rangle . R_1 , \mathbf{1}_2 \langle U_2 \rangle . S_1 \} \\ T_2 &= \mathbf{p}_1? \{ \mathbf{1}_1 \langle U_1 \rangle . R_2 , \mathbf{1}_2 \langle U_2 \rangle . S_2 , \dots , \mathbf{1}_k \langle U_k \rangle . S_k \} \end{aligned}$$

and using (39), (40), and Definitions 4.9 and 4.7, we may verify that:

(i) $G \upharpoonright \mathbf{p}_1 = T_1$, $G \upharpoonright \mathbf{p}_2 \preceq^\sqcup T_2$ and (ii) $\langle\langle T_1 \rangle\rangle = A_1$ and $\langle\langle T_2 \rangle\rangle = A_2$.

- (Case $G = G_1 \mid G_2$): Then $\text{part}(G) = \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$, with $n \geq 2$. Recall that $\text{part}(G_1 \mid G_2) = \text{part}(G_1) \cup \text{part}(G_2)$. By Def. 4.1 we may state the compositional typing assumption as

$$\Gamma; \Delta \vdash \mathbb{M}[G_1] \mid \mathbb{M}[G_2]$$

where $\Delta = c_1:A_1, \dots, c_n:A_n$, with a c_j for each $\mathbf{r}_j \in \text{part}(G_1 \mid G_2)$ (with $j \in \{1, \dots, n\}$). Let $\text{part}(G_1) = \{\mathbf{p}_1, \dots, \mathbf{p}_k\}$ and $\text{part}(G_2) = \{\mathbf{p}_{k+1}, \dots, \mathbf{p}_n\}$. By Prop. B.1, there exist disjoint Δ_1, Δ_2 such that $\Delta = \Delta_1 \cup \Delta_2$ and the compositional typings

$$\begin{aligned} &\Gamma; \underbrace{c_1:A_1, \dots, c_k:A_k}_{\Delta_1} \vdash \mathbb{M}[G_1] \\ &\Gamma; \underbrace{c_{k+1}:A_{k+1}, \dots, c_n:A_n}_{\Delta_2} \vdash \mathbb{M}[G_2] \end{aligned}$$

hold. We may then apply IH on both $\mathbb{M}[G_1]$ and $\mathbb{M}[G_2]$, and so infer local types R_1, \dots, R_k and S_{k+1}, \dots, S_n such that (i) $G_1 \upharpoonright \mathbf{p}_i \preceq^\sqcup R_i$ and (ii) $A_i = \langle\langle R_i \rangle\rangle$ (with $i \in \{1, \dots, k\}$) and (iii) $G_1 \upharpoonright \mathbf{p}_h \preceq^\sqcup S_h$ (iv) $A_h = \langle\langle S_h \rangle\rangle$ (with $h \in \{k+1, \dots, n\}$). This is enough to conclude the thesis, for Def. 3.3 says that parallel global types do not share participants. Therefore, for every $\mathbf{p}_l \in \{\mathbf{p}_1, \dots, \mathbf{p}_n\}$ then either $G \upharpoonright \mathbf{p}_l = G_1 \upharpoonright \mathbf{p}_l$ or $G \upharpoonright \mathbf{p}_l = G_2 \upharpoonright \mathbf{p}_l$.

□

B.4 Proof of Theorem 4.14: Behavioral Characterization of Swapping

We repeat the statement given in Page 8:

Theorem B.8 (4.14). *Let $G_1 \in \mathcal{G}^{\text{fin}}$ be a global type, such that $\mathbb{M}[G_1]$ has a compositional typing $\Gamma; \Delta \vdash \mathbb{M}[G_1]$, for some Γ, Δ . If $G_1 \simeq_{\text{sw}} G_2$ then $\Gamma; \Delta \vdash \mathbb{M}[G_1] \approx \mathbb{M}[G_2]$.*

Proof. We first prove the following property:

$$\text{If } G_1 \simeq_{\text{sw}} G_2 \text{ then } \Gamma; \Delta \vdash \mathbb{M}[G_1] \simeq_c \mathbb{M}[G_2] \quad (49)$$

Recall that we have defined \simeq_c in Def. A.4. The thesis will then follow by combining (49) with Theorem 4.13 (soundness of \simeq_c wrt \approx). The proof of (49) proceeds by induction on the definition of \simeq_{sw} (cf. Def. 4.12 and Fig. 5). We consider only the cases for (sw1) and (sw3):

- (Case (sw3)): Then we have:

$$\begin{aligned} G_1 &= \mathbf{p} \rightarrow \mathbf{q}; \{ \mathbf{1}_i \langle U_i \rangle . (G^i \mid G) \}_{i \in I} \\ G_2 &= G \mid \mathbf{p} \rightarrow \mathbf{q}; \{ \mathbf{1}_i \langle U_i \rangle . G^i \}_{i \in I} \end{aligned}$$

with

$$\{\mathbf{p}, \mathbf{q}\} \# \{\mathbf{r}_1, \dots, \mathbf{r}_n\}. \quad (50)$$

where $\{\mathbf{r}_1, \dots, \mathbf{r}_n\} = \text{part}(G)$. Without loss of generality, we consider the case $I = \{1, 2\}$. Then, by Def. 4.1, we have:

$$\mathbb{M}[G_1] = c_p \triangleright \begin{cases} \mathbf{1}_1 : c_p(u).c_q \triangleleft \mathbf{1}_1; \overline{c_q}(v).([u \leftrightarrow v] \mid \mathbb{M}[G^1] \mid \mathbb{M}[G]) \\ \mathbf{1}_2 : c_p(u).c_q \triangleleft \mathbf{1}_2; \overline{c_q}(v).([u \leftrightarrow v] \mid \mathbb{M}[G^2] \mid \mathbb{M}[G]) \end{cases}$$

Moreover, since by assumption $\mathbb{M}[G_1]$ has a compositional typing, by Theorem 4.10 we have:

$$\Gamma; c_p : \langle\langle G_1 \upharpoonright \mathbf{p} \rangle\rangle, c_q : \langle\langle G_1 \upharpoonright \mathbf{q} \rangle\rangle, c_{\mathbf{r}_1} : \langle\langle G_1 \upharpoonright \mathbf{r}_1 \rangle\rangle, \dots, c_{\mathbf{r}_n} : \langle\langle G_1 \upharpoonright \mathbf{r}_n \rangle\rangle \vdash \mathbb{M}[G_1]$$

Observe that in $\mathbb{M}[G_1]$ we have the name distinctions: $\{c_p, c_q\} \# \{c_{\mathbf{r}_1}, \dots, c_{\mathbf{r}_n}\}$, thanks to (50). These distinctions will be useful in commuting prefixes inside $\mathbb{M}[G_1]$ in a type-preserving way, for they ensure that all these sessions are causally independent. We may now

use \simeq_c to perform the prefix commutations, using the equalities in Figs. 6 and 7:

$$\begin{aligned}
M[G_1] &= c_p \triangleright \left\{ \begin{array}{l} 1_1 : c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1] \mid M[G]) \\ 1_2 : c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2] \mid M[G]) \end{array} \right\} \\
&\simeq_c c_p \triangleright \left\{ \begin{array}{l} 1_1 : c_p(u).c_q \triangleleft 1_1; (M[G] \mid \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1])) \\ 1_2 : c_p(u).c_q \triangleleft 1_2; (M[G] \mid \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2])) \end{array} \right\} \\
&\simeq_c c_p \triangleright \left\{ \begin{array}{l} 1_1 : c_p(u).(M[G] \mid c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1])) \\ 1_2 : c_p(u).(M[G] \mid c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2])) \end{array} \right\} \\
&\simeq_c c_p \triangleright \left\{ \begin{array}{l} 1_1 : M[G] \mid c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \\ 1_2 : M[G] \mid c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2]) \end{array} \right\} \\
&\simeq_c M[G] \mid c_p \triangleright \left\{ \begin{array}{l} 1_1 : c_p(u).c_q \triangleleft 1_1; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^1]) \\ 1_2 : c_p(u).c_q \triangleleft 1_2; \overline{c_q}(v).([u \leftrightarrow v] \mid M[G^2]) \end{array} \right\} \\
&= M[G_2]
\end{aligned}$$

- (Case (SW1)): Then we have:

$$\begin{aligned}
G_1 &= p_1 \rightarrow q_1 : \{1_i \langle U_i \rangle . p_2 \rightarrow q_2 : \{1'_j \langle U'_j \rangle . G_{ij}\}_{j \in J}\}_{i \in I} \\
G_2 &= p_2 \rightarrow q_2 : \{1'_j \langle U'_j \rangle . p_1 \rightarrow q_1 : \{1_i \langle U_i \rangle . G_{ij}\}_{i \in I}\}_{j \in J}
\end{aligned}$$

with $\{p_1, q_1\} \# \{p_2, q_2\}$. Without loss of generality, we consider the following instance:

$$\begin{aligned}
G_1 &= p \rightarrow q : \{1 \langle U_1 \rangle . r \rightarrow s : \{h \langle U_2 \rangle . G\}\} \\
G_2 &= r \rightarrow s : \{h \langle U_2 \rangle . p \rightarrow q : \{1 \langle U_1 \rangle . G\}\}
\end{aligned}$$

with

$$\{p, q\} \# \{r, s\}. \quad (51)$$

Then, by Def. 4.1 on G_1 we have:

$$\begin{aligned}
M[G_1] &= c_p \triangleright \{1 : c_p(u_1).c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid \\
&\quad c_r \triangleright \{h : c_r(u_2).c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid M[G])\})\}
\end{aligned}$$

Moreover, since by assumption $M[G_1]$ has a compositional typing, by Theorem 4.10 we have:

$$\Gamma; c_p : \langle G_1 \upharpoonright p \rangle, c_q : \langle G_1 \upharpoonright q \rangle, c_r : \langle G_1 \upharpoonright r \rangle, c_s : \langle G_1 \upharpoonright s \rangle \vdash M[G_1]$$

Observe that in $M[G_1]$ we have the name distinctions: $\{c_p, c_q\} \# \{c_r, c_s\}$, thanks to (51). These distinctions will be useful in commuting prefixes inside $M[G_1]$ in a type-preserving way, for they ensure that all these sessions are causally independent. We may now use \simeq_c to perform the prefix commutations shown in Fig. 11, using the equalities in Figs. 6 and 7. \square

C. Omitted Proofs from § 4.3

C.1 Characterization Results for Recursive Mediums

This proof follows the lines of that given in Appendix B.2 for Theorem 4.8.

Fact C.1. *Let $G \in \mathcal{G}^\mu$ be a MWF global type. If G has a free variable \mathcal{X} then all the projections of G will have \mathcal{X} as a free variable. Moreover, for all p_i there is a context K_i such that $G \upharpoonright p_i = K_i[\mathcal{X}]$.*

We repeat the statement of Theorem 4.16 (Page 9):

Theorem C.2 (4.16). *Let $G \in \mathcal{G}^\mu$ be a global type, with $\text{part}(G) = \mathcal{P} = \{p_1, \dots, p_n\}$. If G is MWF then*

$$\Gamma; c_1 : \langle G \upharpoonright p_1 \rangle_{\mathcal{P}}^\eta, \dots, c_n : \langle G \upharpoonright p_n \rangle_{\mathcal{P}}^\eta \vdash_\eta M^\mu[G]_k :: k:A$$

is a left compositional typing for $M^\mu[G]_k$ for some Γ, η, A .

Proof. By structural induction on G . We detail only the case $G = \mu\mathcal{X}.G'$, for it is the most interesting case. (Notice that case $G = \mathcal{X}$ does not correspond to a valid global type.) We need to show that, for some Γ, η , and A the following judgment is derivable:

$$\Gamma; c_1 : \langle \mu\mathcal{X}.G' \upharpoonright p_1 \rangle_{\mathcal{P}}^\eta, \dots, c_n : \langle \mu\mathcal{X}.G' \upharpoonright p_n \rangle_{\mathcal{P}}^\eta \vdash_\eta M^\mu[\mu\mathcal{X}.G']_k :: k:A \quad (52)$$

Using the definitions of $\langle \cdot \rangle_{\mathcal{P}}^\eta$, $\langle \cdot \rangle$, and $M^\mu[\cdot]_k$, respectively, the thesis (52) can be equivalently stated as

$$\Gamma; c_1 : \langle \mu\mathcal{X}.G' \upharpoonright p_1 \rangle, \dots, c_n : \langle \mu\mathcal{X}.G' \upharpoonright p_n \rangle \vdash_\eta M^\mu[\mu\mathcal{X}.G']_k :: k:A \quad (53)$$

$$\Gamma; c_1 : \nu\mathcal{X}. \langle G' \upharpoonright p_1 \rangle, \dots, c_n : \nu\mathcal{X}. \langle G' \upharpoonright p_n \rangle \vdash_\eta M^\mu[\mu\mathcal{X}.G']_k :: k:A \quad (54)$$

$$\Gamma; c_1 : \nu\mathcal{X}. \langle G' \upharpoonright p_1 \rangle, \dots, c_n : \nu\mathcal{X}. \langle G' \upharpoonright p_n \rangle \vdash_\eta \text{corec } \mathcal{X}(k).M^\mu[G']_k :: k:A \quad (55)$$

$$\begin{aligned}
M[G_1] &= c_p \triangleright \{1 : c_p(u_1).c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid c_x \triangleright \{h : c_x(u_2).c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid M[G])\})\} \\
&\simeq_c c_p \triangleright \{1 : c_p(u_1).c_q \triangleleft 1; \mathbf{c_x} \triangleright \{h : \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid c_x(u_2).c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid M[G])\})\} \\
&\simeq_c c_p \triangleright \{1 : c_p(u_1).\mathbf{c_x} \triangleright \{h : c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid c_x(u_2).c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid M[G])\})\} \\
&\simeq_c c_p \triangleright \{1 : \mathbf{c_x} \triangleright \{h : c_p(u_1).c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid c_x(u_2).c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid M[G])\})\} \\
&\simeq_c \mathbf{c_x} \triangleright \{h : c_p \triangleright \{1 : c_p(u_1).c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid c_x(u_2).c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid M[G])\})\} \\
&\simeq_c c_x \triangleright \{h : c_p \triangleright \{1 : c_p(u_1).c_q \triangleleft 1; \mathbf{c_x}(u_2).\overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid M[G])\})\} \\
&\simeq_c c_x \triangleright \{h : c_p \triangleright \{1 : c_p(u_1).\mathbf{c_x}(u_2).c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid M[G])\})\} \\
&\simeq_c c_x \triangleright \{h : c_p \triangleright \{1 : \mathbf{c_x}(u_2).c_p(u_1).c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid M[G])\})\} \\
&\simeq_c c_x \triangleright \{h : c_p \triangleright \{1 : c_x(u_2).c_p(u_1).c_q \triangleleft 1; \mathbf{c_s} \triangleleft h; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid M[G])\})\} \\
&\simeq_c c_x \triangleright \{h : c_p \triangleright \{1 : c_x(u_2).c_p(u_1).\mathbf{c_s} \triangleleft h; c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid M[G])\})\} \\
&\simeq_c c_x \triangleright \{h : c_p \triangleright \{1 : c_x(u_2).\mathbf{c_s} \triangleleft h; c_p(u_1).c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid M[G])\})\} \\
&\simeq_c c_x \triangleright \{h : c_p \triangleright \{1 : c_x(u_2).c_s \triangleleft h; c_p(u_1).c_q \triangleleft 1; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid M[G])\})\} \\
&\simeq_c c_x \triangleright \{h : c_p \triangleright \{1 : c_x(u_2).c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid \mathbf{c_q} \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid M[G])\})\} \\
&\simeq_c c_x \triangleright \{h : c_p \triangleright \{1 : c_x(u_2).c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid \mathbf{c_p}(u_1).c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid M[G])\})\} \\
&\simeq_c c_x \triangleright \{h : c_x(u_2).\mathbf{c_p} \triangleright \{1 : c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid c_p(u_1).c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid M[G])\})\} \\
&\simeq_c c_x \triangleright \{h : c_x(u_2).c_s \triangleleft h; \mathbf{c_p} \triangleright \{1 : \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid c_p(u_1).c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid M[G])\})\} \\
&\simeq_c c_x \triangleright \{h : c_x(u_2).c_s \triangleleft h; \overline{c_s}(v_2).([u_2 \leftrightarrow v_2] \mid \mathbf{c_p} \triangleright \{1 : c_p(u_1).c_q \triangleleft 1; \overline{c_q}(v_1).([u_1 \leftrightarrow v_1] \mid M[G])\})\} \\
&= M[G_2]
\end{aligned}$$

Figure 11. Type-preserving process transformations induced by swapping, as required in the proof of Theorem 4.14

We show how to infer (55). First, we record two facts about G' :

$$fv(G') = \{\mathcal{X}\} \quad (56)$$

$$\text{part}(\mu\mathcal{X}.G') = \text{part}(G') \quad (57)$$

Now, by IH the following is a derivable typing judgment, for some η_0, A_0 :

$$\Gamma; c_1 : \langle G' \upharpoonright p_1 \rangle_{\mathcal{P}}^{\eta_0}, \dots, c_n : \langle G' \upharpoonright p_n \rangle_{\mathcal{P}}^{\eta_0} \vdash_{\eta_0} \mathbf{M}^\mu[G']_k :: k : A_0 \quad (58)$$

We describe how (58) allows us to infer (55).

First, using (56) and the definition of $\mathbf{M}^\mu[\cdot]_k$, we infer that $fv(\mathbf{M}^\mu[G']_k) = \{\mathcal{X}\}$. Therefore, since (58) is well-typed, by inversion we may verify that η_0 contains an entry for \mathcal{X} :

$$\eta_0 = \eta'[\mathcal{X}(k) \mapsto \Gamma; \Delta_0 \vdash k : \mathcal{Y}] \quad (59)$$

for some η', Δ_0 . We show that Δ_0 contains an assignment $c_i : B_i$ for all $p_i \in G'$. By assumption G is MWF, and therefore G' is MWF too. As such, for every $p_i \in G'$, the local type $G' \upharpoonright p_i$ is defined. By definition of projection and (56), the recursive variable \mathcal{X} occurs in every B_i . Consequently, by construction, the mapping Δ_0 accounts for all $p_i \in G'$.

A crucial observation is that while the projected type $\langle G' \upharpoonright p_i \rangle$ does have free recursion variables, its corresponding B_i does not. This is induced by typing rule (var). Hence, one of the following holds:

$$\eta_0(\mathcal{X})(c_i) = B_i = \nu\mathcal{X}.\langle G' \upharpoonright p_i \rangle \quad (60)$$

$$\eta_0(\mathcal{X})(c_i) = B_i = \langle G' \upharpoonright p_i \rangle \{ \nu\mathcal{X}.\langle G' \upharpoonright p_i \rangle / \mathcal{X} \} \quad (61)$$

That is, η_0 stores the co-recursive type of the involved projection or its unfolding. We can assume all entries in η_0 are of shape (60), for we may silently rewrite all entries with shape (61) using rule (νL) . Typing inversion ensures that the body of the co-recursive type is the associated projection.

Now, combining (58) with the definition of $\langle \cdot \rangle_{\mathcal{P}}^\eta$ and (60) we infer:

$$\Gamma; c_1 : \langle G' \upharpoonright p_1 \rangle \{ \nu\mathcal{X}.\langle G' \upharpoonright p_1 \rangle / \mathcal{X} \}, \dots, c_n : \langle G' \upharpoonright p_n \rangle \{ \nu\mathcal{X}.\langle G' \upharpoonright p_n \rangle / \mathcal{X} \} \vdash_{\eta_0} \mathbf{M}^\mu[G']_k :: k : A_0 \quad (62)$$

Then, using rule (νL) , we may rewrite (62) as

$$\Gamma; c_1 : \nu\mathcal{X}.\langle G' \upharpoonright p_1 \rangle, \dots, c_n : \nu\mathcal{X}.\langle G' \upharpoonright p_n \rangle \vdash_{\eta_0} \mathbf{M}^\mu[G']_k :: k : A_0 \quad (63)$$

Finally, using rule (νR) , (63), and (59) we may infer

$$\Gamma; c_1 : \nu\mathcal{X}.\langle G' \upharpoonright p_1 \rangle, \dots, c_n : \nu\mathcal{X}.\langle G' \upharpoonright p_n \rangle \vdash_{\eta'} \mathbf{corec} \mathcal{X}(k). \mathbf{M}^\mu[G'] :: k : \nu\mathcal{Y}.A_0 \quad (64)$$

This completes the proof. \square

$$\begin{array}{c}
\frac{c_p : A_i, c_q : B_i, \Delta \vdash \mathcal{M}^\mu \llbracket G_i \rrbracket_k :: k : \langle G_i \rangle}{c_p : A_i, c_q : B_i, q : Q, \Delta \vdash \mathcal{M}^\mu \llbracket G_i \rrbracket_k :: k : \langle G_i \rangle} \text{ (T1L)} \\
\frac{u : U_i \vdash [u \leftrightarrow v] :: v : U_i}{c_p : A_i, c_q : B_i, \Delta \vdash k(q) . \mathcal{M}^\mu \llbracket G_i \rrbracket_k :: k : Q \multimap \langle G_i \rangle} \text{ (T}\multimap\text{R)} \\
\frac{c_p : A_i, u : U_i, c_q : U_i \otimes B_i, \Delta \vdash \overline{c_q}(v) . ([u \leftrightarrow v] \mid k(q) . \mathcal{M}^\mu \llbracket G_i \rrbracket_k) :: k : Q \multimap \langle G_i \rangle}{c_p : A_i, u : U_i, c_q : U_i \otimes B_i, \Delta \vdash k \triangleright \{1_i : \overline{c_q}(v) . ([u \leftrightarrow v] \mid k(q) . \mathcal{M}^\mu \llbracket G_i \rrbracket_k)\}_{\{i\}} :: k : \& \{1_i : Q \multimap \langle G_i \rangle\}_{\{i\}}} \text{ (T}\multimap\text{L)} \\
\frac{c_p : A_i, u : U_i, c_q : \langle G \mid q \rangle, \Delta \vdash c_q \triangleleft 1_i ; k \triangleright \{1_i : \overline{c_q}(v) . ([u \leftrightarrow v] \mid k(q) . \mathcal{M}^\mu \llbracket G_i \rrbracket_k)\}_{\{i\}} :: k : \& \{1_i : Q \multimap \langle G_i \rangle\}_{\{i\}}}{c_p : A_i, u : U_i, c_q : \langle G \mid q \rangle, \Delta \vdash \overline{k}(p) . (\mathbf{0}_p \mid c_q \triangleleft 1_i ; k \triangleright \{1_i : \overline{c_q}(v) . ([u \leftrightarrow v] \mid k(q) . \mathcal{M}^\mu \llbracket G_i \rrbracket_k)\}_{\{i\}}) :: k : P \otimes \& \{1_i : Q \multimap \langle G_i \rangle\}_{\{i\}}} \text{ (T}\otimes\text{R)} \\
\frac{c_p : U_i \otimes A_i, c_q : \langle G \mid q \rangle, \Delta \vdash c_p(u) . \overline{k}(p) . (\mathbf{0}_p \mid c_q \triangleleft 1_i ; k \triangleright \{1_i : \overline{c_q}(v) . ([u \leftrightarrow v] \mid k(q) . \mathcal{M}^\mu \llbracket G_i \rrbracket_k)\}_{\{i\}}) :: k : P \otimes \& \{1_i : Q \multimap \langle G_i \rangle\}_{\{i\}}}{c_p : U_i \otimes A_i, c_q : \langle G \mid q \rangle, \Delta \vdash c_p(u) . \overline{k}(p) . (\mathbf{0}_p \mid c_q \triangleleft 1_i ; k \triangleright \{1_i : \overline{c_q}(v) . ([u \leftrightarrow v] \mid k(q) . \mathcal{M}^\mu \llbracket G_i \rrbracket_k)\}_{\{i\}}) :: k : \langle G \rangle} \text{ (T}\otimes\text{L)} \\
\frac{c_p : U_i \otimes A_i, c_q : \langle G \mid q \rangle, \Delta \vdash k \triangleleft 1_i ; c_p(u) . \overline{k}(p) . (\mathbf{0}_p \mid c_q \triangleleft 1_i ; k \triangleright \{1_i : \overline{c_q}(v) . ([u \leftrightarrow v] \mid k(q) . \mathcal{M}^\mu \llbracket G_i \rrbracket_k)\}_{\{i\}}) :: k : \langle G \rangle}{c_p : \langle G \mid p \rangle, c_q : \langle G \mid q \rangle, \Delta \vdash \mathcal{M}^\mu \llbracket p \multimap q : \{1_i \langle U_i \rangle . G_i \}_{i \in I} \rrbracket_k :: k : \langle G \rangle} \text{ (T}\oplus\text{L)}
\end{array}$$

Figure 12. Typing for Annotated Mediums: Case $G = p \multimap q : \{1_i \langle U_i \rangle . G_i\}_{i \in I}$

We repeat the statement in Page 9

Theorem C.3 (From Well-Typedness To MWF Global Types). *Let $G \in \mathcal{G}^\mu$ be a global type, with $\text{part}(G) = \mathcal{P} = \{p_1, \dots, p_n\}$. If*

$$\Gamma ; c_1 : A_1, \dots, c_n : A_n \vdash_\eta \mathcal{M}^\mu \llbracket G \rrbracket_k :: k : B$$

is a left compositional typing for $\mathcal{M}^\mu \llbracket G \rrbracket_k$ then $\exists T_1, \dots, T_n$ s.t. $G \upharpoonright p_j \preceq^\sqcup T_j$ and $\langle T_j \rangle_p^\eta = A_j$ for all $p_j \in G$.

Proof. By structural induction on G , following the lines of the proof given in Appendix B.3 for Theorem 4.10 (Page 8). \square

C.2 Characterization Results for Annotated Mediums

We repeat the statements in Page 9:

Theorem C.4 (From Well-Formedness To Typed Annotated Mediums). *Let $G \in \mathcal{G}^\mu$ be a global type with $\text{part}(G) = \mathcal{P} = \{p_1, \dots, p_n\}$. If G is WF (Def. 3.4) then judgment*

$$\Gamma ; c_1 : \langle G \upharpoonright p_1 \rangle_p^\eta, \dots, c_n : \langle G \upharpoonright p_n \rangle_p^\eta \vdash_\eta \mathcal{M}^\mu \llbracket G \rrbracket_k :: k : \langle G \rangle$$

is well-typed, for some Γ .

Proof (Sketch). By structural induction on G . The most interesting case is $G = p \multimap q : \{1_i \langle U_i \rangle . G_i\}_{i \in I}$, which follows by extending the argument detailed in Appendix B.2 for the proof of Theorem 4.8, using the typing derivation in Figure 12, where we omit Γ and Δ stands for the typing of all $p_j \in G_i$. \square

Theorem C.5 (From Well-Typed Annotated Mediums To WF Global Types). *Let $G \in \mathcal{G}^\mu$ be a global type. If the following judgment is well-typed*

$$\Gamma ; c_1 : A_1, \dots, c_n : A_n \vdash \mathcal{M}^\mu \llbracket G \rrbracket_k :: k : A_0$$

then $\langle G \rangle \preceq^\oplus A_0$ and $\exists T_1, \dots, T_n$ s.t. $G \upharpoonright r_j \preceq^\sqcup T_j$ and $\langle T_j \rangle_p^\eta = A_j$ for all $r_j \in G$.

Proof. By structural induction on G . \square

C.3 Operational Correspondence via Annotated Mediums

The following property follows directly from the definition of annotated mediums (Definition 4.4) and systems (Definition 4.22) as well as from the properties of typed processes.

Proposition C.6. *Let G be a MWF global type, and let $\mathcal{S}^k(G)$ be the systems implementing G . We have:*

1. *for all $P_j \in \mathcal{S}^k(G)$, we have $\Gamma ; \cdot \vdash P_j :: k : \langle G \rangle$, for some Γ .*
2. *for all $P_j \in \mathcal{S}^k(G)$, we have that if $P_j \xrightarrow{\lambda} P'$ with $\lambda \neq \tau$ then $\text{subj}(\lambda) = k$.*

We repeat the statement given in Page 10:

Theorem C.7 (Global Types and Mediums: Operational Correspondence). *Let G be a MWF global type and P any process in $\mathcal{S}^k(G)$. We have:*

- (1) *If $G \xrightarrow{\sigma} G'$ then there exist λ, P' such that $P \xrightarrow{\lambda} P'$, $\lambda = \llbracket \sigma \rrbracket^k$, and $P' \in \mathcal{S}^k(G')$.*
- (2) *If there is some P_0 such that $P \Longrightarrow P_0 \xrightarrow{\lambda} P'$ with $\lambda \neq \tau$ then there exist σ, G' such that $G \xrightarrow{\sigma} G'$, $\text{subj}(\sigma) = p$, $\sigma = \llbracket \lambda \rrbracket_p$, and $P' \in \mathcal{S}^k(G')$.*

Proof. Part (2) is easy, relying on Proposition C.6 and on the definitions of system and annotated mediums. We notice that the (finite) reduction sequence leading to P_0 (i.e., preceding the observable action on λ) must necessarily involve at least one synchronization between the annotated medium and its closure. As for part (1), we consider only the case in which the global transition is realized via rule

$$(G1) \quad p \rightarrow q: \{1_i \langle U_i \rangle . G_i\}_{i \in I} \xrightarrow{\overline{p \triangleleft 1_j}} p \rightsquigarrow q: 1_j \langle U_j \rangle . G_j \quad (j \in I)$$

since other cases are similar. Writing p_1 and p_2 instead of p and q , by Definition 4.22 we have:

$$P = (\nu c_{p_1}, \dots, c_{p_n})(Q_{p_1} \mid Q^* \mid \mathcal{M}^\mu \llbracket p_1 \rightarrow p_2: \{1_i \langle U_i \rangle . G_i\}_{i \in I} \rrbracket_k)$$

where $Q^* = Q_{p_2} \mid \dots \mid Q_{p_n}$ and for all $i \in \{1, \dots, n\}$ Definition 4.21 ensures that

$$\Gamma; \cdot \vdash Q_{p_i} :: c_{p_i}: \langle\langle G \mid p_i \rangle\rangle$$

In particular, well-typedness of $\mathcal{M}^\mu \llbracket p_1 \rightarrow p_2: \{1_i \langle U_i \rangle . G_i\}_{i \in I} \rrbracket_k$ ensures that

$$\Gamma; \cdot \vdash Q_{p_1} :: c_{p_1}: \oplus \{1_i : U_i \otimes B_i\}_{i \in I}$$

We now have a weak transition (which is finite, due to Theorem 3.7(3)):

$$P \implies (\nu c_{p_1}, \dots, c_{p_n})(Q'_{p_1} \mid \dots \mid Q_1^* \mid \mathcal{M}^\mu \llbracket p_1 \rightarrow p_2: \{1_i \langle U_i \rangle . G_i\}_{i \in I} \rrbracket_k) = P_1$$

where typing preservation (Theorem 3.7(1)) ensures that $Q'_{p_1} \xrightarrow{\overline{c_{p_1} \triangleleft 1_j}} Q''_{p_1}$. Then, Definition 4.4 ensures a synchronization on name c_{p_1} ; by expanding the definition of annotated medium we obtain:

$$\begin{aligned} P_1 &\xrightarrow{\tau} (\nu c_{p_1}, \dots, c_{p_n})(Q''_{p_1} \mid \dots \mid Q_1^* \mid k \triangleleft 1_j; \mathcal{M}^\mu \llbracket p_1 \rightsquigarrow p_2: 1_j \langle U_j \rangle . G_j \rrbracket_k) \\ &\xrightarrow{\overline{k \triangleleft 1_j}} (\nu c_{p_1}, \dots, c_{p_n})(Q''_{p_1} \mid \dots \mid Q_1^* \mid \mathcal{M}^\mu \llbracket p_1 \rightsquigarrow p_2: 1_j \langle U_j \rangle . G_j \rrbracket_k) = P' \end{aligned}$$

and by Definition we have that $\llbracket \overline{p \triangleleft 1} \rrbracket^k = \overline{k \triangleleft 1}$. Finally, we observe that by performing a selection action, we obtain that the offer of Q''_{p_1} evolves: $\Gamma; \cdot \vdash Q''_{p_1} :: c_{p_1}: U_j \otimes B_j$. Therefore, by expanding the definition of system, we may infer $P' \in \mathcal{S}^k(p_1 \rightsquigarrow p_2: 1_j \langle U_j \rangle . G_j)$. \square